Claus-Jochen Haake

Friedhelm Meyer auf der Heide

Marco Platzner

Henning Wachsmuth

Heike Wehrheim (Eds.)

On-The-Fly Computing

Individualized IT-services in dynamic markets

Collaborative Research Centre 901 (2011 – 2023)

Preface

The provision of complex IT services is a challenging task that needs expertise from a variety of areas. Today, a lot of support is provided, for example by architectural concepts such as microservices, by novel implementation concepts such as container virtualization for the separation, scaling, orchestration and management of resources, or by providers of frameworks, pre-fabricated components or Cloud services. Nevertheless, combining such supporting basic services to a desired IT service constitutes a challenge that goes beyond the capabilities of even a very experienced, ambitious expert. Instead, a team of experts is necessary in order to develop and deploy a complex IT service that comprises a multitude of different components running on different platforms. One way out could be to (at least partially) automate the process of configuring the IT service out of given basic services. Examining this approach and demonstrating its feasibility is the topic of the Collaborative Research Centre (CRC) 901 "On-The-Fly Computing".

This monograph presents overviews and highlights of our research over the entire funding period (July 2011 – June 2023) of the CRC 901. It consists of reports from the subprojects as well as from our transfer projects. Note that we only report about projects that still run during the third funding period. That is why Subprojects A2 (research is continued in C4) and C3 (key researcher is retired) are not listed. Two transfer projects, T3 and T4, have just started. Therefore, they are described just with short abstracts in this monograph, not in great detail.

In our proposal for this CRC 901, written in 2011, we formulated a vision that describes such an approach to the provision of IT services:

"Our vision of On-The-Fly Computing is that of IT services that are individually and automatically configured and brought to execution from flexibly combinable services traded on markets. At the same time, we aim to organize markets whose participants maintain a vibrant market of services through appropriate entrepreneurial action."

Our research is based on the following structure of On-The-Fly Computing markets (OTF markets henceforth) (see Figure 1): Customers (users) formulate their desired IT services and send them to OTF providers who have expertise in the domain from which the requested service originates. These providers compete for a contract with the customer and create a configuration using the software and execution services offered in the market by various types of service providers (supplier). Software services are offered by OTF software providers and execution services – i.e., the timely execution of the configured software – are offered by OTF compute centers. In this context, customers, OTF providers, OTF software providers and OTF compute centers act as entrepreneurs in the OTF market. Consequently, the realization of our vision necessitates developing methods for service quality assurance and the protection of participating clients and providers, methods for the target-oriented further development of markets, and methods to analyze the participants' incentives and to support their interaction in dynamically changing markets.

The research agenda requires expertise from various areas of computer science and economics. The CRC combines research foci such as computer networks and distributed algorithms, security and cryptography, software engineering and verification, configuration
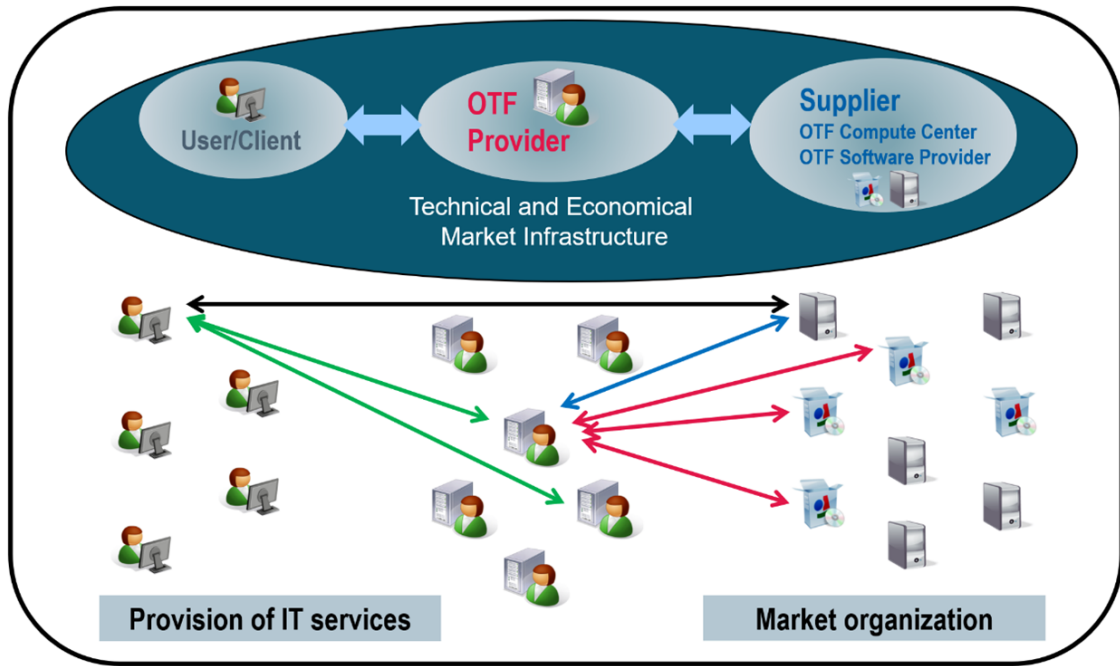
Figure 1: *Structure of the OTF markets.*

and machine learning, computer engineering and HPC, microeconomics and game theory, and business computing and management.

Our CRC is divided into four project areas. **Project Area A** examines *Algorithmic and Economic Foundations* for organizing large dynamic markets. **Project Area B** investigates processes for *Modeling, Composition and Quality Analysis* of services and service configurations, aiming at on-the-fly development of high-quality IT services. **Project Area C** develops *Reliable Execution Environments and Application Scenarios* for On-The-Fly Computing and is concerned with questions of the stability and security of markets, the organization of highly heterogeneous OTF compute centers, and the provision of configured services by those centers. In addition, it develops a framework for OTF market platform architectures as well as their software-technical realization. **Project Area T** bundles the *transfer projects* of our CRC, which provide a framework for joint research by our CRC researchers and external partners and facilitate the exchange of results from applied research back into our fundamental research.

Project Area A: Algorithmic and Economic Foundations for the Organization of Large, Dynamic Markets.

The task of this project area is to support the interaction of the participants in the OTF market from the technical side and to organize it from the economic side in such a way that inefficiencies are avoided. To this end, we have a close look at characteristics of OTF markets and take account of the dynamics and heterogeneity of the market participants, possibilities for regulating their interaction, and the dis-/satisfaction collected through valuation systems.

In **Subproject A1:** *Possibilities and Limits of Local Strategies in Dynamic Networks*, we view the actors of our OTF markets as peers in a peer-to-peer system, where an

overlay network supports the interaction. We explore how local methods can be used to adapt the overlay network to changes caused, for example, by a changing membership or changing requirements of the applications. In order to address different types of changes, we focus on self-stabilizing overlays, i.e., overlays that can recover from any state, scalable distributed data structures, and hybrid networks, which are networks supporting different communication modes. In such dynamic environments, we also study variants of resource allocation problems such as resource leasing or heterogeneity of resources.

In **Subproject A3:** *The Market for Services: Incentives, Algorithms, Implementation*, we analyze the OTF market for composite services from a microeconomic perspective providing flexible ways to organize the interaction in a dynamic environment. In particular, we use models and techniques from cooperative, non-cooperative and algorithmic game theory. We mainly address the different forms of interaction including: a) bargaining problems, when only few participants are involved, b) matching models as a form to design the rules of interaction, and c) models of competition, which cover the interaction of whole groups in the OTF market and allows an evaluation of the market outcome from the perspective of social welfare.

In **Subproject A4:** *Empirical Analysis in Markets for OTF Services*, we analyze how quality signals of customer reviews and certifications can reduce information asymmetries and contribute to a well-functioning online market. We use data from existing electronic markets and conduct laboratory or online experiments to identify factors that impact the generation, provision or attributes of customer reviews, analyze the economic impact of customer reviews, and examine design decisions for online review systems. We also evaluate the effectiveness of certification beyond customer ratings to reveal true service quality.

Project Area B: Modeling, Composition and Quality Analysis

This project area investigates description, composition, and analysis techniques for single services and service compositions. The goal is to achieve a precise yet simple description of services and requirements, which then allow an automatic configuration of service compositions. The quality of the service composition in terms of functional as well as non-functional requirements is achieved through innovative verification and certification methods as well as machine learning techniques.

Subproject B1: *Parameterized Service Specification* aims to facilitate end-user participation in the configuration process by supporting natural language requirements specifications without limiting expressiveness. This presented a number of challenges, such as the ambiguity of natural language statements or the need for a successful interaction between the customer and the software component during specification. For this reason, a chatbot with extensive compensation and interaction capabilities was developed. New computational methods were brought up that encode requirement-specific information into natural language explanations while adjusting the style to the customer's proficiency.

Subproject B2: *Configuration and Evaluation* develops methods and algorithms for the configuration and evaluation of software services in the OTF computing scenario. Starting with formal specifications of functional requirements and the use of techniques from automated planning for service composition, the focus continuously shifted toward the use of machine learning (ML) methods for adapting and improving the service composition

process in a data-driven way. At the same time, machine learning served as an important use case: Motivated by recent work on automated machine learning (AutoML), OTF machine learning considers the provision of ML functionality, an important and practically relevant type of service in the context of OTF markets.

Subproject B3: *Composition Analysis in Uncertain Contexts* researches analysis methods for service compositions that evaluate the fulfilments of functional and non-functional requirements. Throughout the lifetime of the CRC, Subproject B3 has investigated various requirements, ranging from functional requirements described by logical pre- and postconditions or protocol specifications to non-functional requirements such as memory consumption. As Subproject B3 has studied service compositions assembled by Subproject B2, a specific focus in later periods has been on the analysis of compositions of machine learning services.

In **Subproject B4:** *Proof-Carrying Services*, we investigate techniques for ensuring the quality of services that are procured on the market and assembled on-the-fly, without trusting the service provider. We aim at developing methods and concepts that allow the consumer of a service to automatically, formally and quickly check if the service possesses the desired properties, i.e., if it adheres to a given specification. To this end, we developed various forms of analysis (for both hardware and software services) that provide information about the validity of properties and that can help customers to check properties. In the end, such analysis can be augmented to become full proof-carrying software or hardware methods.

Project Area C: Reliable Execution Environments and Application Scenarios

This project area develops reliable execution environments for the On-The-Fly Computing and is concerned with questions of the robustness and security of markets, the organization of high-grade heterogeneous OTF Compute Centers and the execution of configured services by such centers. In addition, we aim at developing an architecture framework for OTF markets.

Subproject C1: *Robustness and Security* develops methods and techniques that ensure high robustness and security in OTF markets. Concerning robustness, we consider information systems as well as overlay networks. We are interested in methods that protect against denial-of-service (DoS) attacks, even against insiders, and that mitigate the effects of such attacks, i.e., information systems that remain available in the presence of DoS attacks. Concerning security, we focus on techniques that guarantee strong authentication of data and entities while also preserving user privacy. To achieve this we construct enhanced signature schemes and anonymous credentials. Finally, we study secure and anonymous reputation systems. Such systems help actors in OTF markets to find appropriate services.

In **Subproject C2:** *On-The-Fly Compute Centers I: Heterogeneous Execution Environments*, we investigate the execution of configured IT services in OTF compute centers with heterogeneous compute nodes, which use CPUs, GPUs and FPGAs as compute resources. Since each service can exist in multiple variants that differ in their non-functional properties (e.g., latency, throughput, energy consumption) depending on the used computing and communication resources, the operation of OTF compute centers needs to be considered as a multi-objective optimization problem with constraints. We have developed models for

characterizing composed services and execution architectures and validated them with simulation but also with empirical evaluation in prototypical testbeds to quantify the benefits of heterogeneous computing resources and networks.

In **Subproject C4:** *On-The-Fly Compute Centers II: Execution of Composed Services in Configurable Compute Centers*, we are concerned with efficiently utilizing resources within highly configurable compute centers. We consider services at different levels of abstraction, from network services to data-parallel applications. In addition, we focus on properties of composed services that are typical for OTF computing. This subproject emphasizes the collaboration between theoretical and practical computer science on closely related issues. We examine these issues by using various methods (theoretical analysis, simulation, emulation, prototyping) on different levels of abstraction.

In **Subproject C5:** *Architectural Management of OTF Computing Markets*, we investigate structural decisions and dynamic business behavior to design commercially successful OTF computing markets. Based on empirical and conceptual research methods, we obtain the following three major outcomes: First, we study literature and comparative markets to derive an architectural framework for OTF computing markets. Second, we analyze business model development methods, modeling languages, and software tools and develop our own artifacts for creating ecosystem-oriented business models. Third, we conduct an exploratory interview study with potential stakeholders to derive success factors of a future OTF market.

Project Area T: Transfer Projects

Transfer projects serve to verify the results of the fundamental research carried out in our CRC under practical conditions or to enable the development of prototypes. They also provide a framework for joint research by our CRC researchers and external partners and facilitate the exchange of results from applied research back into our fundamental research.

In **Transfer Project T1:** *Flexible Industrial Analytics on Reconfigurable Systems-on-Chip*, we deal with mapping industrial analytics functions to embedded heterogeneous compute platforms. Industrial analytics is a current trend in automation and denotes capturing and analyzing a multitude of measurements from machines and production processes to create added value for future operation. Together with the application partner Weidmüller Interface GmbH & Co. KG we work on reconfigurable System-on-Chip (rSoC) technology with the goal to optimize implementations by assigning functions to software and hardware at both design time and runtime. The challenges in using rSoC for industrial analytics are to provide the required flexibility for system design while mastering the increasing heterogeneity of rSoC platforms. The developed technology is demonstrated in case studies such as condition monitoring for wind turbines and welding machines.

In **Transfer Project T2:** *Practicable Cryptographic Techniques for Secure and Data-efficient Customer Loyalty Systems*, a new privacy-preserving customer loyalty system for the retail sector is being developed and evaluated. In a customer loyalty system, customers are rewarded for retail purchases. Often these systems award a certain number of points for each euro spent. Currently implemented systems typically require customers to reveal lots of private data such as their purchase history. Together with Diebold Nixdorf – a service provider for the retail sector – and building upon research done in Subproject C1,

the transfer project realizes a prototype system to prove that privacy-preserving loyalty systems can be built, to showcase the power of the cryptographic techniques, and to evaluate whether such a system is suited for real-world use.

Transfer Project T3: *Automated Risk Analysis with Respect to Open-source Dependencies (Hektor)* aims to research, develop and assess novel techniques to efficiently and precisely detect and mitigate the inclusion of known-to-be-vulnerable third-party dependencies within software compositions. The project seeks to build an open-source toolchain called HEKTOR, which will support the secure development of applications and services at a massive scale. In collaboration with SAP – a world leader for the development and provision of cloud services for business-to-business applications – we aim to implement and evaluate HEKTOR such that it is ready to be applied on a large scale and to a large and diverse set of real-world software development projects. The focus of HEKTOR lies especially on the detection of modified software dependencies that have been re-compiled or re-bundled. In addition to the detection capabilities, HEKTOR will perform a reachability analysis and determine whether the vulnerable code is executable at all. Based on the results of the reachability analysis HEKTOR, can remove unreachable code and therefore minimize the possible attack surface of the application.

Transfer Project T4: *Online Reviews on B2B Platforms for Software Products* builds primarily on the results of fundamental research carried out under Subproject A4. We aim to identify the underlying motives for writing online reviews in B2B markets. Unlike online reviews in B2C markets, these motives are little understood and largely unexplored for B2B markets. Given the systematic differences between B2C and B2B markets, and based on the literature and anecdotal evidence, we posit that these motives are not the same. We expect some motives known from B2C to be less significant or even non-existent in a B2B context, and, above all, to uncover B2B-specific motives, too. In this transfer project, we will characterize and distinguish between online reviews in these two types of markets, and we use a multi-method approach to identify and validate the underlying motives for writing B2B online reviews in software markets, specifically. For this reason, we have teamed up with CELONIS, a global business software that acts as a service company. The insights from this transfer project will enable B2B companies to elicit and exploit B2B online reviews in a more targeted manner.

<div align="right">

June 2023
Claus-Jochen Haake
Friedhelm Meyer auf der Heide
Marco Platzner
Henning Wachsmuth
Heike Wehrheim

</div>

Contents

Subproject A1:

Capabilities and Limitations of Local Strategies in Dynamic Networks

Thorsten Götte[1], Till Knollmann[2], Friedhelm Meyer auf der Heide[2],
Christian Scheideler[1], Julian Werthmann[1]

1 Department of Computer Science, Paderborn University,
Paderborn, Germany
2 Heinz Nixdorf Institute and Department of Computer
Science, Paderborn University, Paderborn, Germany

1   Introduction

The On-The-Fly (OTF) market communication infrastructure plays a central role in our envisioned OTF ecosystem. At its core, it must ensure that the participants can communicate efficiently and reliably. During our efforts, we are facing many fundamental problems and challenges: in particular because the size and the dynamics of these systems makes it unrealistic to control and optimize them using classical centralized strategies executed by an entity that has full information about the current state of the system. Therefore, we explored the capabilities and limitations of local algorithms. An algorithm is *local* if each node in a distributed system only needs to interact with its neighbors in order to solve a given task. In large distributed systems, these neighborhoods continuously change, which will also affect the communication.

An especially challenging aspect is that of *external dynamics*, i.e., the network changes due to events that are outside of its control. External dynamics can happen for a variety of reasons. Most notably, they can be caused by attacks, faults, or changes in the membership. However, since the participants operate under ever changing economic or legal conditions, they might re-evaluate their communication interests and therefore change their interaction patterns.

Once the market reaches a certain size, centralized management approaches do not work anymore for handling external dynamics efficiently. For instance, a centralized controller that serves as an entry point to the market can quickly become a bottleneck or even a single point of failure if many actors try to join at once. Further, a malicious controller could even influence the market in its favor by denying certain providers from entering the market. Hence, distributed and ideally local strategies are needed. The goal of these strategies should not just be to maintain a fixed topology but to also adapt the topology to the needs of the market participants.

One of the topics that we focused on was self-stabilizing overlay networks, i.e., networks that can recover their topology from any weakly connected state. Many overlay networks

thgoette@mail.upb.de (Thorsten Götte), tillk@mail.upb.de (Till Knollmann), fmadh@upb.de (Friedhelm Meyer auf der Heide), scheideler@upb.de (Christian Scheideler), jwerth@mail.upb.de (Julian Werthmann)

have been proposed before, among them pioneering works such as Chord, CAN, Pastry, and Tapestry. However, these were not self-stabilizing. Later, various self-stabilizing overlay networks have been proposed, including self-stabilizing lists, skip graphs, de Bruijn graphs and Delaunay graphs (see, e.g., [FSS21] for an overview of stabilizing and non-stabilizing networks). However, they do not necessarily ensure monotonic self-stabilization in a sense that all parts of the network that are already in a desired state will remain in a desired state during self-stabilization. Since this property is critical for a high availability, we particularly focused on monotonically self-stabilizing overlay networks. An overview of our solutions in given in **Section 2.1**. *Monotone Searchability: A New Paradigm for Self-stabilizing Algorithms*.

Developing scalable solutions for distributed data structures was another topic of our research. Many solutions have already been proposed for distributed hash tables before our research, but no solutions were known before that scale well for inherently sequential data structures such as stacks, queues, and heaps. We did not just come up with highly scalable solutions that work under the very general asynchronous message passing model. We also devised ones that ensure strict consistency requirements such as sequential consistency. We describe our contributions in **Section 2.2**. *Distributed Data Structures: Scalability Meets Consistency*.

During the last years we initiated the study of hybrid communication networks. These are networks that support two different communication modes: a local mode that only allows the nodes to exchange information with some fixed, predetermined neighborhood, and a global mode where the nodes are able to change the network topology over time arbitrarily. Hybrid networks have already been used in various contexts such as data centers in which servers exchange information via some wired infrastructure as well as wireless or optical communication. Another prominent example is that of hybrid multi-point VPNs, where connections between their participants are established via leased lines as well as best-effort connections via TCP/IP. Despite their importance in practice, theoretical research on hybrid networks did not exist before our work. We particularly focused on solving graph problems via hybrid networks, such as finding a minimum spanning tree or shortest paths in the local network. We present more details of this research in **Section 2.3**. *Hybrid Networks: Exploiting the Heterogeneity of Modern Communication Infrastructures*.

Besides our research about models of networks and their dynamics, we have investigated resource allocation problems: Where should resources in a network be placed so that both the cost for placement and access are minimized? A prominent example is the facility location problem. In this problem, we are given a space with locations and distances between them, a function defining the cost of opening a facility at each location, and a sequence of requests at locations. Every request of the sequence must be served by a facility open when the request arrives for a price of the distance between the request and the respective facility. The goal is to open up facilities so as to minimize the total opening cost plus the total cost for serving all requests. Our contributions extend previous work in several directions: We developed efficient approximation algorithms in a dynamic settings by distributed algorithms. Further, we considered the online case where requests are not known in advance but instead arrive over time. An algorithm here needs to serve arriving requests immediately while not knowing future ones. Among others, we have analyzed leasing approaches, where resources are not bought and then exist "forever" but can be leased for different time periods and for different prices. We have investigated

the potential of allowing resources to be moved in the network (mobile resources), and we have generalized facility location and other resource allocation problems to take the heterogeneity of the resources into account. We describe our contribution to resource allocation in **Section 2.4**. *Online Allocation of Resources: Providing Services Without Knowledge on Future Demands*.

2 Main Contributions

In the following, we summarize the main contributions of our subproject.

2.1 Monotone Searchability: A New Paradigm for Self-Stabilizing Algorithms

Like any large-scale distributed system, our OTF market infrastructure will undergo frequent changes during its operation. The exact nature of these changes can be manifold. On the one hand, there are benign changes of the membership, e.g., participants joining or leaving the market in a coordinated way, or changes in the communication structure. On the other hand, there might also be actively harmful changes caused by malicious attackers. These attacks could even be aided by insiders in the market that benefit from failures of their competitors. For example, ill-intentioned service providers could try to exclude their competitors from the composition algorithm. They can achieve this by attacking the network in a way such that the corresponding composition requests do not reach other providers anymore and their services are not considered. Therefore, a key requirement for a robust OTF market infrastructure is *resilience* against these attacks as long as this is possible and *fast recovery* once the attack is over. In this subproject, we focused on solutions for fast recovery. As the global state of the system after a failure or attack might be arbitrarily corrupted, our protocols must potentially be able to recover the infrastructure from scratch. In particular, we want our system to reach a desired configuration from *any* initial configuration in a *finite* number of steps. This paradigm is commonly referred to as *self-stabilization* and has sparked a vast amount of research in the last 50 years. In his seminal paper, Esger W. Dijkstra defined self-stabilization as follows:

Definition 1 (Self-stabilization (cf. [Dij74])) *Let $\mathfrak{C}$ be the set of all possible states of a system. Then, a protocol is self-stabilizing w.r.t. to a set of legitimate states $\mathcal{L} \subseteq \mathfrak{C}$ if it satisfies the following two properties.*

- *Convergence: starting from an arbitrary system state $S_0 \in \mathfrak{C}$, the protocol is guaranteed to arrive at a state $S_t \in \mathcal{L}$ in a finite number of steps.*

- *Closure: starting from a legitimate state the protocol remains in legitimate states thereafter.*

A self-stabilizing protocol is thus able to recover from transient faults regardless of their nature. Moreover, a self-stabilizing protocol does not have to be initialized as it eventually starts to behave correctly regardless of its initial state.

For our particular case, we model the system as dynamic graph, more precisely, as an overlay network. Each node in the graph represents a market participant and each edge

represents a TCP/IP connection between the participants. We assume that nodes can delegate their edges by sending the corresponding IP addresses. Thus, a state might be modeled by a set of nodes V, their internal variables, and the messages in transit to each node. A legitimate state might then include a particular graph topology or a family of graph topologies. For each state S_t, we can define a graph $G(S_t) := G_t := (V, E_t^e \cup E_t^i)$ with two kinds of edges. First, there are *explicit* edges E_t^e with $(u, v) \in E_t^e$ if and only if u stores a reference to v in its local memory in step t. Second, we call an edge $(u, v) \in E_t^i$ *implicit* if and only if there is a reference to v in the channel of u in step t (and will eventually be received). Arguably, the most important operation in this model is the so-called *delegation*. Here, a node sends of its stored references to another node and thereby creates a new implicit edge.

It is well understood how one can build a plethora of useful network topologies in this model by delegating the edges. This research area is commonly referred to topological self-stabilization and it has seen a large number of impactful results in the last two decades. The investigated topologies range from simple line graphs, over sophisticated topologies of low degree and diameter, to spanners for a given metric space. A recent survey paper provides a comprehensive overview of the state-of-the-art [FSS21] of the techniques and algorithms for topological self-stabilization. Many of these protocols were developed during the first funding period of the CRC and/or by key researchers of this subproject. However, the specifics of these protocols are not the focus here. Instead, we consider a more structural problem shared by all these protocols: None of them provide any guarantees about the stabilization process (other than its eventual convergence). In particular, some desired functionalities or properties that hold in some configuration $S_\tau \notin \mathcal{L}$ may be violated in later state $S_{\tau'}$ with $\tau' > \tau$ that can be reached be the protocol. We argue that in many cases this is not the desired behavior of a resilient protocol. Intuitively, we want a stabilization protocol to *improve* in every time step, i.e., once parts of the system are operational, they should remain operational to ensure the best possible service.

One example of such functionality is *searching*, i.e., the ability to route a message to a specific node in the system. This is arguably one of the most frequently used functionality of any distributed system and — in the context of the CRC — crucial for the composition process. If we can find a node $v \in V$ from another node $w \in V$ in some configuration S_t, we want to be able to do so in all subsequent configurations. In this case, we say the protocol satisfies *monotone searchability*

Seeking a formal definition of the problem, let us first clarify what we mean by *searching*. We consider search requests, i.e. , $\texttt{search}(v, destID)$ messages that are routed according to a given routing protocol R, where v is the sender of the message and $destID$ is the identifier of a node we are looking for. Note that $destID$ does not necessarily belong to an existing node w but rather represents a virtual address that may or may not be occupied by a node. A search request can either *succeed* or *fail*. We say it *succeeds* if a $\texttt{search}(v, destID)$ message reaches a node w with $id(w) = destID$. Otherwise, if the message reaches some node u with $id(u) \neq destID$ **and** cannot be forwarded anymore according to R, the search request *fails*. We assume that nodes themselves initiate $\texttt{search}()$ requests at will. Given these preliminary thoughts and definitions, we formally define *monotone searchability* as follows:

Definition 2 (Monotone Searchability (cf. [SSS15])) *We say a (self-stabilizing) proto-*

col P with legitimate states $\mathcal{L}$ satisfies monotonic searchability *according to some routing protocol R if it holds for any pair of nodes v, w that once a* search$(v, id(w))$ *request (that is routed according to R) initiated at time t succeeds, any* search$(v, id(w))$ *request initiated at a time $t' > t$ will succeed.*

In the following, we will slightly abuse notation and refer to a tuple $(\mathcal{P}, \mathcal{L}, R)$ of a self-stabilizing protocol $\mathcal{P}$, its set of legitimate states $\mathcal{L}$, and a routing algorithm R simply as a *protocol*. The research on monotone searchability went in two directions. First, the goal was to find efficient protocols that satisfy monotone searchability given a concrete set of legitimate states $\mathcal{L}$ and a concrete routing algorithm R. In this area, our researchers could develop protocols for constructing a sorted list, a skip list, and quadtrees embedded in the Euclidean plane. For more details on these protocols, we again refer to the aforementioned survey [FSS21]. The second (and arguably more interesting) direction considered the question of which properties must be fulfilled by a protocol to support monotone searchability, or, on a related note, which classes of protocols can be transformed into protocols that satisfy monotone searchability. Surprisingly, in [SSS16] Setzer, Scheideler, and Strothmann could show that a broad class of existing self-stabilizing protocols containing virtually all protocols developed in the CRC can be transformed to satisfy monotonic searchability. In the remainder of this section, we will quickly outline their approach. We begin by considering the two preconditions a protocol must fulfill to be suitable for the transformation, a generic distance metric dist$(\cdot, \cdot)$ and the so-called MDL property.

Distance oracle dist$(\cdot, \cdot)$: First, all nodes need to have access to a distance oracle dist $: V \times V \to \mathbb{R}$ that takes two identifiers as input and output their distance in some shortest path metric. Further, the nodes must be able to evaluate the oracle locally. While this might sound like a hard and unhandy assumption at first, a closer look into existing protocols reveals this is indeed fulfilled by almost all of them. For example, many topologies contain a sorted list of its nodes. Assuming that identifiers are integers in $[1, n]$, one can simply define dist$((v, w)) := |v - w|$ and obtain an oracle with all desired properties.

MDL property: The next property (or rather set of properties) is dubbed MDL property in [SSS16]. It encapsulates four basic features that ensure that a protocol gets monotonically closer to the legitimate states. More precisely, a deterministic protocol $(\mathcal{P}, \mathcal{L}, R)$ fulfills the MDL property if for any action a of the protocol it holds that:

1. An edge $(u, v) \in E_{\mathcal{L}}$ will **never** be delegated by $u \in V$.

2. If an edge $(u, v) \notin E_{\mathcal{L}}$ is delegated in step τ, it will be delegated in all steps $\tau' > \tau$ (if u receives another reference of v).

3. Any stable edge that may be traversed by the search protocol (and any implicit edge that results from it delegation) is only delegated to a node whose distance (in the underlying distance metric) is closer to the target than the current node.

Informally speaking, the first two properties imply that the protocol *monotonically* converges to its desired topology, since edges of the topology are always kept and edges that are not part of the topology are obviated over time. The last property states that the delegation of edges can only improve the routing. Further, if the legitimate states contain implicit edges, we additionally require the following:

4. In every legitimate state, for any implicit edge (u, v), there are fixed cycle-free paths $(u = u_1, u_2, \ldots, u_k)$ such that u_i sends the reference of v to u_{i+1}, and u_k has an explicit edge (u_k, v), i.e., the reference of v is forwarded along fixed paths until it finally fuses with an existing reference.

This property implies that all implicit edges will eventually merge with explicit ones in legitimate states. Note that the MDL property is generally not a severe restriction as almost all topological self-stabilization algorithm fulfill it naturally.

Given all these two preconditions and constructions, the main result was the following:

Theorem 1 (Main Result of [SSS16]) *Any self-stabilizing protocol* $(\mathcal{P}, \mathcal{L}, R)$ *that satisfies the* MDL *property can be turned into a protocol* $(\mathcal{P}', \mathcal{L}, R')$ *that satisfies monotone searchability.*

The actual construction of Setzer, Scheideler, and Strothmann has two building blocks. First, they adapt the mechanisms how edges are delegated by the protocol. Second, they introduce a generic routing protocol that exploits the distance metric and the MDL property. Again, we give a brief overview over both approaches.

Safe delegations: Recall that our model does not assume FIFO delivery, so messages may be received out of sequence. This creates problems if a node v forwards a reference to another node and a $\texttt{search}(\cdot, \cdot)$ message that needs to be sent to that reference. With non-FIFO delivery, the $\texttt{search}(\cdot, \cdot)$ message could *overtake* the reference and the search fails although it worked when v still stored the reference. Thus, our protocols need to cope with the non-FIFO message delivery. To address this issue, Setzer, Scheideler, and Strothmann developed a set of extensions for the standard topology manipulation primitives. On a high level, these extended primitives ensure that edges delegated away by some node $v \in V$ are not integral to the routing. Their key technique to achieve this is to *warn* neighboring nodes that there will be a delegation and then wait for their acknowledgment that this delegation is indeed safe. This, however, requires the use of sequence numbers to match warnings and their acknowledgments. Thus, the system only stabilizes if the initial state does not contain corrupted sequence numbers. Although this goes against the main idea of self-stabilization, it is (in some sense) unavoidable. The authors proved that under non-FIFO message delivery, no protocol could maintain monotone searchability from arbitrary initial states. Therefore, either the initial set of states must be restricted or one needs to assume FIFO delivery.

Generic routing: Finally, the authors need to account for the fact that the routing R takes paths that are not guaranteed to persist. To cope with this problem, they introduce a *generic search protocol R'* that exploits the distance oracle $\texttt{dist}(\cdot, \cdot)$ and the MDL property. Informally speaking, the message always takes the *smallest possible* step towards the target with respect to $\texttt{dist}(v, w)$ that it has seen so far. To this end, all identifiers *seen* along the path are also stored in the message. If the search gets stuck in some node, it can use these to *backtrack* and try another path.

2.2 Distributed Data Structures: Scalability Meets Consistency

Like in the sequential world, efficient distributed data structures are important in order to realize efficient distributed applications. The most prominent type of distributed data

structure is the distributed hash table (DHT). Many distributed data stores employ some form of DHT for lookup. Important applications include file sharing (e.g., BitTorrent), distributed file systems (e.g., PAST), publish-subscribe systems (e.g., SCRIBE), and distributed databases (e.g., Apache Cassandra). However, other distributed forms of well-known data structures, such as queues, stacks, and heaps, have received much less attention although queues, for example, have a number of interesting applications as well. The main challenge of coming up with scalable distributed solutions of queues, stacks, and heaps is that they are inherently sequential, i.e., the challenge is to execute the operations in a distributed fashion so that the outcome of the execution is consistent with a sequential execution. We considered the following forms of consistency.

Definition 3 *Let $\mathcal{DS}$ be a distributed data structure on a node set V and let OP be set of requests issued to the data structure. Further, each request $op(u, i) \in OP$ is associated with a node $u \in V$ and sequence number $i \in \mathbb{N}$. Then, we say*

1. *$\mathcal{DS}$ is **serializable** if and only if there exists an ordering $\prec$ on the set OP so that the distributed execution of all requests in OP on the data structure is equivalent to the serial execution w.r.t. $\prec$.*

2. *$\mathcal{DS}$ is **locally consistent** if and only if there exists an ordering $\prec$ on the set S so that for all u and i it holds that $op(u, i) \prec op(u, i + 1)$.*

3. *$\mathcal{DS}$ is **sequentially consistent** if and only if it is serializable and locally consistent w.r.t. the same ordering.*

Intuitively, local consistency means that for each node v, the requests issued by v have to come up in $\prec$ in the order they were issued by that node. Thus, it is somewhat independent of the data structure's semantics and may always be achieved by a trivial lexicographic order. The same cannot be said of serializability, which is greatly affected by the data structure's semantics and, moreover, is highly non-trivial to be achieved efficiently in a distributed system.

For example, consider the classical (sequential) stack data structure where `Push(i)` adds a data item i and `Pop()` returns the data item that was added last. In the distributed setting, each node is able to invoke `Push(i)` and `Pop()`. The order relation $\prec$ must ensure that the request can mapped to a sequential execution. In particular, this means that any `Pop()` operation that returns i is preceded by a `Push(i)` operation (by some node). Moreover, there must be no `Push(j)` operation with $i \neq j$ ordered in between the operation (unless there is another `Pop()` that removes `j`). A straw-man distributed solution could simply see one node responsible for the stack and let it handle all `Push(i)` and `Pop()` requests. However, this would hardly be scalable as a single node needs to process the entire system's traffic. Thus, an efficient solutions must carefully distribute the responsibilities for each requests while still ensuring serializability. The feasibility greatly depends on the data structure's concrete semantics. Besides the stack, we were able to develop solutions for several different well-known and established data structures, which at first glance might seem inherently sequential. These are presented in the remainder of this chapter.

Distributed queue: A distributed queue can be used to come up with a unique ordering of messages, transactions or jobs, and it can be used to realize fair work stealing since tasks available in the system would be fetched in FIFO order. Other applications are distributed

mutual exclusion, distributed counting, or distributed implementations of synchronization primitives. Server-based approaches of realizing a queue in a distributed system already exist, such as Apache ActiveMQ, IBM MQ, or JMS queues. Many other implementations of message and job queues can be found at http://queues.io/. However, none of these implementations provides a queue that allows massively parallel accesses without requiring powerful servers. The major problem of coming up with a fully distributed version of a queue is that its semantics are inherently sequential. Nevertheless, we were able to come up with a distributed protocol for a queue ensuring sequential consistency that fairly distributes the communication and storage load among all members of the distributed system and that can efficiently process even massive amounts of `Enqueue()` and `Dequeue()` requests. Our protocol works in the asynchronous message passing model and can also handle massive amounts of join and leave requests efficiently.

A distributed queue has to implement four operations: `Enqueue()`, `Dequeue()`, `Join()` and `Leave()`. `Enqueue()` adds an element to the queue and `Dequeue()` removes an element from the queue so that the FIFO requirement is satisfied. `Join()` allows a process to enter the system while `Leave()` allows a process to leave the system.

We presented a distributed queue ensuring sequential consistency under the asynchronous message passing model, which also ensures a high scalability. More precisely, when assuming synchronous message passing, our `Enqueue()` and `Dequeue()` operations are processed in $O(\log n)$ communication rounds w.h.p., where n is the number of nodes. Furthermore, we show that we can process n `Join()` or $n/2$ `Leave()` operations in $O(\log n)$ rounds, w.h.p. Through the use of a distributed hash table, our distributed queue allocates its elements equally among all processes such that no process stores significantly more elements than the rest [FSS18a]. In the arXiv version of our paper, we also showed how to use the techniques for our distributed queue to come up with a highly scalable distributed stack ensuring sequential consistency [FSS18b].

Distributed priority queue: We also presented a highly scalable distributed priority queue (or heap). A distributed heap might be useful in scheduling, for example, where one might insert jobs that have been assigned priorities and workers might pull these jobs from the heap based on their priority. Another application for a distributed heap is distributed sorting. A distributed heap supports the following operations:

- `Insert(e,p(e))`: Inserts the element `e` with priority `p(e)` into the heap.
- `DeleteMin()`: Retrieves the element with minimum priority from the heap or returns $\perp$ if the heap is empty.
- `Join()`: The node v issuing the operation wants to join the system.
- `Leave()`: The node v issuing the operation wants to leave the system.

We distinguished between settings that only allow a constant amount of priorities and settings for arbitrary amounts and presented two novel distributed protocols for these scenarios – Skeap and Seap. Both protocols support insertions and deletions of elements in time $O(\log n)$ w.h.p., where n is the number of processes participating in the heap. Furthermore, we provided some guarantees on the semantics, by having Skeap guarantee sequential consistency and Seap guarantee serializability. For part of Seap, we obtained a novel protocol KSelect for distributed k-selection that runs in $O(\log n)$ rounds w.h.p. Both Skeap and Seap work in the asynchronous message passing model. To provide an

additional feature we can handle join and leave requests of processes in time $O(\log n)$ w.h.p. without violating the heap semantics or losing important data. Even though Seap comes with slightly weaker semantics than Skeap, it only uses $O(\log n)$ bit messages for its operations, while the message size in Skeap partially depends on the rate with which processes generate new operations [FS19].

2.3 Hybrid Networks: Exploiting the Heterogeneity of Modern Communication Infrastructures

In our study of hybrid networks, we consider systems with multiple modes of communication. Specifically, we allow our nodes to employ *local* communication with a fixed set of neighbors and global communication with any node in the network. We find many examples of such networks in the context of the CRC. Consider, for example, an execution of a composed service. Since the composition consists of multiple services from multiple providers that possibly have restrictions on where and how their services are initiated, it is unlikely to be executed by single compute node at a single location. Rather, it involves several servers at multiple compute centers at different geographical locations. Such a setup may be seen as a hybrid network. A single server can easily access data stored by servers in the same compute center as are they equipped with capable networking hardware. These would be the local connections, which can be used for large data transfers. On the other hand, large-scale data transmissions between different compute centers are possible in principle, but much slower. Therefore, a connection with a server in another center can be seen as a global connection. Due to their limitations, these global connections should rather be used for metadata, control messages, or intermediate results. Moreover, there are several avenues for hybrid networks beyond the scope of the CRC. For example, they can also be motivated by the ability of modern smart phones to employ device-to-device communication (local) as well as their internet connection (global). As the latter is usually connected to a cost, we tend to impose harsher limits on the number of messages each device is allowed to send via this connection in a given amount of time.

In our research, we focused on efficient algorithms for these hybrid networks that cleverly exploit these different communication capabilities. In the remainder of this section, we will present a more formal treatment of the model and give an overview of our and related results.

The Hybrid communication model (cf. [AHK$^+$20]): In the HYBRID communication model, we consider a fixed set of nodes V with identifiers of length $O(\log n)$. Time is synchronous, i.e., divided into rounds. At the beginning of a round, each node receives messages sent to it in the last round. Afterwards it may perform an arbitrary amount of local computation and then send messages to other nodes that will be delivered in the next round. Specifically, these nodes can communicate in two modes and we parameterize the model by the messaging capacities allowed in each of them. For the so-called local mode, we are given a fixed set of edges and each node may send λ messages of size $O(\log n)$ to each of its neighbors in every round of communication. For the so-called global mode, we assume the nodes to be connected as a clique. However, each node may only send and receive γ messages of size $O(\log n)$ in total in every round of communication.

Special non-hybrid cases of the HYBRID model are the LOCAL model ($\lambda = \infty, \gamma = 0$),

the CONGEST model ($\lambda = O(1)$, $\gamma = 0$) and the node-capacitated clique (NCC, $\lambda = 0$, $\gamma = O(\log n)$). In some cases, a more restricted version of the model is studied, where initially no global edges exist. In this case, each node may send any identifier of a node it knows via a message to have the target node learn about the node in the message, thereby establishing a global edge between them. This model of global communication is called NCC_0 for $\gamma = O(\log n)$. Sometimes λ and γ denote the amount of bits that are allowed to be sent over local or global edges, respectively. Finally, we allow the edges of the HYBRID model to be weighted by a weight function w assigning a natural numbered weight to each of the local edges.

Hybrid shortest paths algorithms: An interesting branch of our research on hybrid networks focused on the investigation of shortest paths algorithms. We initiated this research with the paper *Shortest Paths in a Hybrid Network model* by Augustine et al. ([AHK+20]), where we studied how the addition of global edges affects the runtime of SSSP and APSP compared to classical model. To this end, the very powerful LOCAL model was picked for the local edges and the rather restrictive NCC was picked for the global edges, i.e., $\lambda = \infty$ (though $\lambda = n$ suffices for our algorithms) and $\gamma = O(1)$. The main contributions are presented in Table 1 for APSP and Table 2 for SSSP. We want to specifically mention the APSP lower bound of $\widetilde{\Omega}(\sqrt{n})$ for $\widetilde{O}(\sqrt{n})$-approximations. Additionally, we stress that the runtimes beat the non-hybrid lower bound of $\Omega(D)$ for many graphs.

APSP	Approx	Weights	Complexity	Local Capacity
	Exact	✓	$\widetilde{O}(n^{2/3})$	$O(n)$
	$(1 + \varepsilon)$	-	$\widetilde{O}(\sqrt{n/\varepsilon})$	$O(n)$
	3	✓	$\widetilde{O}(\sqrt{n})$	$O(n)$
	$\widetilde{O}(\sqrt{n})$	-	$\widetilde{\Omega}(\sqrt{n})$	∞

Table 1: *Overview of the APSP contributions of [AHK+20].*

SSSP	Approx	Weights	Complexity	Local Capacity
	Exact	✓	$\widetilde{O}(\sqrt{\text{SPD}})$	$\widetilde{O}(n^2)$
	$(1 + \varepsilon)$	✓	$\widetilde{O}(n^{1/3}/\varepsilon^6)$	$\widetilde{O}(n^{2/3}\varepsilon^6)$
	$(1/\varepsilon)^{O(1/\varepsilon)}$	✓	$\widetilde{O}(n^\varepsilon)$	$O(1)$
	$2^{O\left(\sqrt{\log n \log \log n}\right)}$	✓	$2^{O\left(\sqrt{\log n \log \log n}\right)}$	$O(1)$

Table 2: *Overview of the SSSP contributions of [AHK+20].*

In the paper *Fast Hybrid Network Algorithms for Shortest Paths in Sparse Graphs* by Feldmann et al. ([FHS20]), we further study the SSSP problem in graphs with only a few edges. Here, we parameterize the hybrid model with $\lambda = O(1)$ and $\gamma = O(\log n)$, restricting the local edges to the CONGEST model. Hence, both local and global is limited, resulting in a more realistic setting. We present deterministic $O(\log n)$ time SSSP algorithms for path graphs, cycle graphs, trees and pseudotrees (i.e., graphs with exactly one cycle). Additionally, we present a randomized $O(\log n)$ time SSSP algorithm for cactus graphs (i.e., graphs where any two cycles share at most one node) and a randomized $O(\log^2 n)$ time algorithm for any graph with at most $n + O(n^{1/3})$ edges and arboricity $O(\log n)$ (cf.

Table 3). As the lower bound for the CONGEST model is $\Omega(D)$, the logarithmic runtimes correspond to an exponential speedup caused by the addition of global edges. In addition to this, the paper presents many useful techniques for HYBRID algorithms such as multiple aggregation techniques and a technique allowing the simulation of algorithms for the well-established PRAM model.

SSSP	Class	Approx	Weights	Complexity
	Path Graphs	Exact	✓	$O(\log n)$
	Cycle Graphs	Exact	✓	$O(\log n)$
	Trees	Exact	✓	$O(\log n)$
	Pseudotrees	Exact	✓	$O(\log n)$
	Cactus Graphs	Exact	✓	$O(\log n)$ w.h.p.
	∗	3	✓	$O(\log^2 n)$ w.h.p.

∗: $n + O(n^{1/3})$ edges and arboricity $O(\log n)$

Table 3: *Overview of the contributions of [FHS20].*

In the paper *Near-Shortest Path Routing in Hybrid Communication Networks* by Coy et al ([CCF+22]) and its successor paper [CCS+23], we shift our attention from SSSP to routing. Specifically, we consider how we can construct routing tables and node labels of size $O(\log n)$ that allow us to forward packets from a source to a target such that the path taken is worse than the shortest path by a constant factor only. We restrict our research on *unit disk graphs* (UDGs) that have nodes embedded in $\mathbb{R}^2$ and an edge between two nodes iff their Euclidean distance is at most 1. To this end, we show how to transform any routing scheme for grid graphs, i.e., graphs with nodes embedded in $\mathbb{Z}^2$ and edges between nodes that have distance of 1, to a routing scheme for UDGs in constant time while only losing a constant factor in the distances traveled. This allows us to present an exact routing scheme for grid graphs and obtain the desired routing scheme for UDGs.

Resulting related work: Our research on shortest paths in the HYBRID model sparked interesting research by other authors that has been published at established and competitive conferences. In their paper *Distance Computations in the Hybrid Network Model via Oracle Simulations* Censor-Hillel, Leitersdorf, and Polosukhin provide an exact weighted shortest path algorithm that runs in $\tilde{O}(n^{1/3})$ rounds, w.h.p. [CLP21]. To achieve this, they exploit the ability of nodes with many local connections to distribute data much quicker than nodes that are not as well connected. Specifically, their goal is to simulate *oracles* that are able to receive $\deg(v)$ messages from each node v per simulated round. Note that this would enable them to learn the entire graph, which is known to be difficult in HYBRID and would therefore require a large overhead in runtime. Hence, they restrict the simulation to a skeleton graph roughly representing the entire graph. This allows the simulation of one round of oracle communication in $\tilde{O}(n^{1/3})$ rounds, w.h.p. Further, the paper *Routing Schemes and Distance Oracles in the Hybrid Model* by Schneider and Kuhn [KS22] investigated the lower bounds in the HYBRID model. The authors show that constant factor approximations of APSP require a runtime polynomial in n on general graphs. More precisely, they present a worst-case graph where computing an α-approximation of all shortest paths takes $\Omega(n^{O(1/\alpha)})$ time. This shows that our results on general graphs cannot fundamentally be improved to, say, polylogarithmic runtimes (unless $\alpha \in \Omega(\log(n))$). Further, the lower bound of [KS22] is accompanied by algorithms that (almost) match this

bound.

Outlook: Although [CLP21] and [KS22] provide strong results, this does not mean that the topic of (approximate) APSP is exhausted. Recall that the lower bound from [KS22] only holds for general graphs as the worst-case instance has a quite unique and pathological topology. As a consequence, we will focus our attention on restricted graph classes that often appear in practice. We have already made some progress here, as evidenced by [FHS20] and [CCF⁺22], but strive to extend this with even more practically motivated graphs classes such as planar graphs or graphs embedded in the Euclidean plane.

2.4 Online Allocation of Resources: Providing Services without Knowledge on Future Demands

Within the scenario of our CRC not only are the clients interested in an automatic composition of software services, they also desire to use the created services. To this end, services are executed in a distributed system and the system itself should manage how and where the services are deployed while the clients access them. Nowadays, deployment of software in distributed systems is done virtually such that the services can be managed widely independent of the physical infrastructure. One common approach is to run services in virtual machines that simulate a physical machine by software. The main advantage of such virtual machines for our system is that they allow us to manage the deployed services dynamically. For example, they can be migrated (moved to another participant) if one participant receives too many requests, or re-configured if the specifications of services or requirements of clients change. Our system benefits from these properties a lot as it further allows us to design algorithms that optimize the placement of the services.

Resource allocation problems: On the one hand, clients are interested in having their desired services at network participants that are close, as this reduces the delay while utilizing the services. On the other hand, the deployment of services itself requires time, energy, and available local resources (computation time, memory), so we should, for example, avoid deploying too many copies of services to ensure that our system scales. Such a setting can be modeled by the well-established field of research around *resource allocation* problems. Here, a set of *resources* (virtual machines containing services) is managed by an algorithm to serve a sequence of *requests*. The goal is to minimize a cost function. Costs are given by the actions of the algorithm such as deployment cost or migration cost and also by the cost to serve requests (for example, given by the distance of a request to the nearest resource).

Within the broad area, there are several models for resource allocation that consider different possible scenarios. The facility location problem, for example, considers the setting where the algorithm is only allowed to deploy new resources and serves requests by connecting them to the nearest one. Deploying and connecting incurs the cost here. Other instances of resource allocation problems are the file migration problem, the k-server problem, and the set cover problem. In file migration, the algorithm cannot deploy a new resource but it can migrate the existing one. Similarly, the algorithm cannot deploy new resources but migrate existing ones in the k-server problem. In comparison to file migration, the algorithm must place one copy (of k existing ones) on the location of each arriving request to serve it (it cannot serve a request by connecting it). In the set cover

problem, each request asks for an element of a given ground set. The algorithm serves a request by offering the requested element, which can be accomplished by buying subsets of the ground set. These subsets are also fixed and given as part of the input.

Approximation algorithms: Many resource allocation problems are difficult to compute. The facility location problem, for example, is known to be NP-hard and motivates the research on *approximation algorithms*. Here, the aim is to compute an approximate solution having at most p times the cost of the optimal solution, where p is the *approximation factor*. In turn, the computation time of such an approximate solution is polynomial in the input. For example, one of the early approximation algorithms for the facility location problem achieves an approximation factor of 3.16 [STA97]. In our research, we concentrated on distributed approximation algorithms for resource allocation problems. On top of the distributed setting, we considered a network that is under *external dynamics*, i.e., that changes over time uncontrollably. In [ACD+11], we presented two *local* approximation algorithms for the metric facility location problem. Both have a poly-logarithmic runtime in the number of peers and a constant approximation factor (one of the two algorithms is slightly faster than the other but may provide a slightly worse solution). It is especially interesting that both algorithms can deal very well with the aforementioned external dynamics.

Online algorithms: Besides the difficulty to compute solutions even when all requests are known, resource allocation offers the following additional challenge. On top of the task to manage services optimally, the requests of clients are usually not known beforehand in our system. In contrast, they arrive over time and future requests are usually unknown. Therefore, the main difficulty that our system has to deal with is maintaining a good placement of the services while adapting the placement for future unknown requests. This motivates us to consider resource allocation problems in a setting where requests arrive *over time* while the algorithm has to take irrevocable actions to guarantee that each request gets served before the next arrives. Such problems are denoted as *online problems*. The criterion of actions being *irrevocable* captures a natural rule: When the algorithm decides to execute an action, the cost of it is paid immediately. The decision cannot later be taken back to reduce the cost when knowledge about future requests is obtained. Of course, an algorithm following such a strong restriction might not be able to perform as well as one that knows all requests beforehand. To measure the loss in performance due to considering a problem online, the *competitive ratio* was introduced and evolved into a standard measure (as amortized efficiency in [ST85]):

Definition 4 (Competitive Ratio) *Let P be a problem with a set of instances I. Let* Alg *be an online algorithm and* Opt *be an optimal offline algorithm for P. Denote by* $\text{Cost}(A, i)$ *the total cost of an algorithm A on an instance $i \in I$. Then* Alg *is called c-competitive if for all instances $i \in I$ for some constant a independent of i it holds that*

$$\text{Cost}(\textsc{Alg}, i) \leq c \cdot \text{Cost}(\textsc{Opt}, i) + a.$$

The competitive ratio allows us to measure how well an algorithm (Alg) performs in terms of the best way it could perform had it known all requests beforehand (Opt). Note that the ratio is a worst-case ratio, i.e., it expresses a guarantee of the performance of an online algorithm regarding every possible valid input sequence for the respective problem.

As an example, consider the facility location problem mentioned already above. Here, the lower bound on the competitive ratio is $\Omega(\frac{\log n}{\log \log n})$ where n is the number of arriving requests [Fot08]. On the other hand, there are several algorithms with a competitive ratio close to this bound, e.g., a randomized algorithm by Meyerson with an expected competitive ratio of $O(\frac{\log n}{\log \log n})$ [Mey01; Fot08]. Regarding online resource allocation, we considered three directions; leasing of resources, mobility, and heterogeneity.

Leasing problems: In *leasing*, the classical problems are extended by assuming that each resource that is otherwise bought is now *leased* for a fixed time only. The possible lease lengths and their costs are given as part of the input. When the lease for a resource runs out, it has to be leased again if needed. Leasing captures the intuition that a deployment of a service should not remain forever but is only required as long as clients using the service arrive. The leasing model above became famous as the parking permit problem considered by Meyerson [Mey05], which itself generalizes the ski rental problem. Regarding the parking permit problem, Meyerson showed a deterministic lower bound of $\Omega(k)$ and a randomized lower bound of $\Omega(\log k)$ where k is the number of available leases. For both cases — randomized and deterministic — asymptotically optimal algorithms are known. The parking permit problem grasps the difficulty of leasing very precisely and has been applied to other resource allocation problems to extend them. For example, Nagarajan and Williamson used it to extend the aforementioned online facility location problem by leasing in [NW13]. Here, requests can only be served by resources available at the point in time the request arrives. They presented an algorithm achieving a competitive ratio of $O(k \log n)$, i.e., with an additional factor of k in the competitive ratio due to the leasing in comparison to the classic facility location problem.

Leasing introduces the additional difficulty of not only deciding on *where* and *when* resources are deployed, but also *how long* they are leased. In [KMP12; AKM$^+$16], we presented the first algorithm for online facility leasing that is independent of the length of the request sequence n. In general, the algorithm has a competitive ratio of $O(\ell_{\max} \log \ell_{\max})$ where $\ell_{\max}$ represents the length of the longest lease. Additionally, for many natural input sequences where the number of arriving requests per time step does not vary too much in consecutive steps, the algorithm achieves a better competitive ratio of $O(\log^2 \ell_{\max})$. Leasing can also be applied to other resource allocation problems such as the set cover problem. The leasing variant of the set cover problem has previously been studied exclusively as an offline problem (cf. [AG07]). Here, the model extension is similar to the online facility location problem, i.e., sets are no longer bought but have to be leased following the given available leases. We have presented the first online algorithms for set cover leasing in [AMM14]. Our randomized algorithms also work for the variant of set cover with repetitions presented by Alon et al. [AAG09], where elements appear multiple times and must be served by a different set each time. Our results improve their competitive ratio of $O(\log^2(m \cdot n))$ to $O(\log m \, \log(m \cdot n))$, where n is the size of the ground set and m is the number of available sets that can be leased.

Mobile resource augmentation: Regarding *mobility*, we have addressed the question of whether and, if so, to what extent we can improve resource allocation through resource mobility. This involves a model that allows moving resources in Euclidean space to make them more usable. Our models capture the intuition that a slight adaption of the location a service is deployed at has a negligible cost, while it might significantly improve the overall cost. As a basis, we have considered a simple model [FM19] similar to the

file migration problem, in which a single resource can be moved over a finite distance at each time step to better serve future requests. Recapitulate that in the file migration problem [BS89], the movement of the algorithm is unlimited although the cost increases with the moved distance. If both the optimal solution and the online algorithm are allowed to move resources, the competitive ratio is immediately dependent on the length of the request sequence. Therefore, we consider mobility as a *resource augmentation*. Resource augmentation gives additional actions to the online algorithm while still comparing its performance to an optimal solution that does not have these actions.

For mobility, this means that the online algorithm is allowed to move its resources in a very limited way while the optimal solution cannot move its resources. For the base problem mentioned above, we have shown that a simple algorithm is sufficient to achieve an optimal competitive factor. In [FKMM21], we extended the model to deal with k mobile resources. Here, the additional decision an algorithm has to make is which resource to move. In general, no online algorithm can even have a bounded competitive ratio in our model. To achieve a bounded competitive ratio, one has to restrict the power of the adversary by enforcing *locality of requests*, i.e., consecutive requests are only allowed to arrive at a bounded distance to the previous request. We present a general algorithm with a bounded competitive ratio in this setting in [FKMM21].

Further, we applied mobility to the facility location problem [FKM22]. More in detail, we allow an online algorithm to move opened facilities between serving requests at an appropriate cost. We compare the cost of such an algorithm with an optimal solution that knows all requests in advance but generates a static solution in which the facilities are not shifted. In [FKM22], we showed that the lower bound for the classical online facility location problem of $\Omega(\frac{\log n}{\log\log n})$ [Fot08] can be beaten in this model. The lower bound for the classical model implies a dependence on the competitive factor on the number of requests. We were able to remove this dependence completely, leaving the competitive factor to depend only on the parameters of the cost function. Our results are asymptotically optimal.

Heterogeneous requests and resources: Our most recent focus lies on *heterogeneity*, i.e., systems in which the offered resources are not all the same but offer different commodities. Requests in such a system can specify which commodities they are interested in as well. Our extensions capture that the set of services (commodities) managed in our system is heterogeneous, while a joint management often implies lower costs compared to managing each kind of service on its own. For example, deploying multiple different services (commodities) at the same location is usually cheaper than deploying them all on their own, as the common deployment overhead is only paid once. One problem we generalized by commodities is the online facility location problem in [CFK$^+$20]. Formerly, the multi-commodity facility location problem has only been researched in the offline case [RS04]. Here, whenever the algorithm deploys a resource, it has to decide on which commodities to offer at the resource. The offered commodities influence the deployment cost such that offering a combination of commodities in one facility is cheaper than offering the same set of commodities by multiple facilities. Here, the additional difficulty for an online algorithm is to decide on which set of commodities to offer when constructing a facility. In general, our model extension increases the competitive ratio in the lower bound by an additive $\sqrt{|S|}$, where S is the set of commodities. Intuitively, the bound comes from the observation that no online algorithm can efficiently guess the correct set of commodities that is required

by future requests even at a single point. On the algorithmic side, we presented online algorithms, a deterministic and a randomized one, that gain at most an additional factor of $\sqrt{|S|}$ in the competitive ratio for many cost functions. The increase in the competitive ratio however heavily depends on the function describing the cost of the algorithm.

Heterogeneous k-server problems: Another perspective on heterogeneity in resource allocation can be seen in [CFK+22], where we extended the k-server problem. Next, we would like to go into more technical details regarding this publication. The k-server problem was first introduced in 1988 [MMS88], where a lower bound of k for deterministic algorithms on uniform metrics was shown. In uniform metrics, the k-server problem reduces to the paging problem. It was famously conjectured (and not proven up to today) that there is a deterministic algorithm achieving a competitive ratio of k for any metric. In our work in [CFK+22], we assume that the servers are all distinct and any arriving request can either demand to be served by any server (*general requests*) or a specified one (*specific requests*). This extension yields some interesting effects. One might be tempted to assume that the competitive ratio decreases when specific requests appear because a request leaving no choice on which server has to move can be trivially served. Especially, when all requests are specific, a competitive ratio of 1 is easy to achieve by simply moving the servers as dictated by the input. Perhaps surprisingly, instances in which both general and specific requests arrive yield a higher lower bound of $2k - 1$.

The Lower bound: To see this, consider the following instance on a uniform metric with $k + 1$ many locations $v_1, \ldots, v_{k+1}$ and any deterministic online algorithm. Let $s_1, \ldots, s_k$ be the algorithm's servers and $o_1, \ldots, o_k$ be the optimal ones. Initially, assume that $s_i = o_i = v_i$ for all $1 \leq i \leq k$, i.e., the algorithm's servers are exactly at the same position as the optimal ones. Since the algorithm acts deterministic, we can assume that in the following sequence the algorithm moves its servers in the order of the indices. Our input sequence uses general requests at the location that is not covered by the algorithm until the algorithm covers the locations $v_1, \ldots, v_{k-1}, v_{k+1}$. Since the algorithm moves its servers in order, this looks as follows: The first request appears at v_{k+1} and the algorithm moves s_1 there. Then, the next request appears at v_1, and the algorithm either moves s_1 back or it moves s_2 there. Due to the deterministic behavior of the algorithm, there is an ordering of the servers such that it moves them in the order of the indices and hence has k movements until it covers all positions except for v_k. Essentially, the optimal solution only needs to move o_k to v_{k+1} and is done. If we stop the sequence here, we have the original lower bound of k for the classical k-server problem. However, due to specific requests, we can increase the cost of the algorithm even further. Note how the optimal solution has all servers o_i at v_i for $1 \leq i \leq k - 1$. Therefore, the optimal solution does not have to move if we simply add specific requests for all these servers at their initial positions. The algorithm however moves all these servers. Either it already moved a server i twice, or it has to move i due to the specific request back to its initial position. Therefore, for all servers except s_k, the algorithm has a movement cost of at least 2. In total, this yields a cost of the algorithm of $2k - 1$ while the optimal solution only needs one movement.

Intuitively, the lower bound shows that by heterogeneity it is no longer sufficient to cover the same positions as the optimal solution. Further, an online algorithm must converge the location of each of its servers to the matching server in the optimal solution.

A Trade-off: The lower bound we just described uses $2k - 1$ requests in total, k of them being general requests and $k - 1$ of them being specific ones. Consider the ratio of

specific requests requiring a movement vs. all requests requiring a movement. We only consider requests requiring a movement of the algorithm, as all others do not influence the competitive ratio. For the lower bound above, the ratio is $\frac{k-1}{2k-1} \approx \frac{1}{2}$. When there are only general requests, the ratio is 0 and the lower bound is only k. On the other hand, when there are only specific requests, the ratio is 1 and the lower bound is simply 1. In our work, we noticed that the ratio plays an important role in the competitive ratio. Therefore, we established a more general lower bound parameterized in the aforementioned ratio. To see a plot of it, consider Figure 2 below, where s expresses the ratio. As we can see, the starting from k at $s = 0$, the lower bound increases up to roughly $\frac{1}{2}$. Thereafter it very quickly decreases to 1 for $s = 1$.
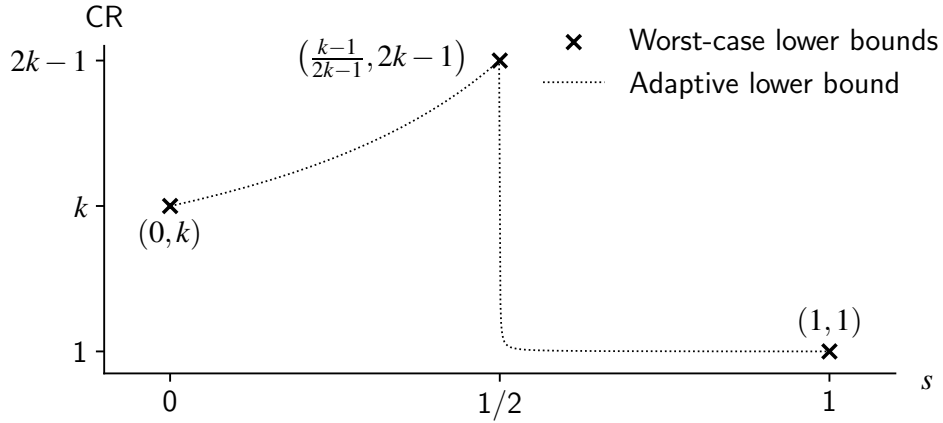
Figure 2: *A plot of our adaptive lower bound of [CFK$^+$22]. CR is the competitive ratio and s is the ratio between the number of specific requests requiring a movement and the total number of requests requiring a movement.*

This hints at the following: The adversary in our model only has significant power just until specific requests start to dominate the input sequence. To complement this, we showed algorithms for the uniform metric space that have a tight competitive ratio for $s > \frac{1}{2}$. So, the interesting part in the competitive ratio happens for $s < \frac{k-1}{2k-1}$. Here, the question arises if we can find an algorithm with a tight competitive ratio. We show that no such algorithm can exist. More specifically, no algorithm can have a competitive ratio of k for $s = 0$ and $2k - 1$ for $s \approx \frac{1}{2}$. This is because to achieve a good worst-case competitive ratio close to $2k - 1$, an algorithm has to follow a specific behavioral rule that we establish in [CFK$^+$22]. If an algorithm does so, we can show a lower bound in the case of $s = 0$, raising the competitive ratio by one for each server following the rule. If one does not follow the rule, a competitive ratio of k can be achieved for $s = 0$. However, each server not acting according to the rule raises the competitive ratio by one in the worst case.

What we see here is a trade-off that would remain hidden if we did not parameterize in s. In advance, we have to choose if we would like to have a good performance on classical k-server instances, or in the worst case, when specific requests appear. The good news is that we presented two algorithms following this trade-off. The one is optimal for $s = 0$ and achieves a worst-case competitive ratio of $3k - 2$, while the other is close to optimal in the worst-case with a competitive ratio of $2k + 14$ but has at least a competitive ratio of $2k - 1$ if $s = 0$. Both algorithms can be mixed to derive an algorithm that has a fine-tuned competitive ratio dependent on s.

Our research on online resource allocation explored different aspects of the field. We focused on leasing approaches, where resources are leased for a limited time instead of being bought permanently. We then enhanced online algorithms by incorporating limited mobility to test the boundaries when minor adjustments are permitted. Lastly, we expanded classical problems to accommodate heterogeneity, enabling requests and resources to handle multiple commodities.

3 Concluding Remarks

Our research of Subproject A1 of our CRC has significantly extended the state-of-the-art in the area of overlay networks, hybrid networks, and resource allocation problems. We envision hybrid approaches to be an important direction for future research. One particularly interesting direction seems to be hybrid wireless networks. In a hybrid wireless network, the participants can establish local connections via Wi-Fi connections and can also establish arbitrary global connections via some given infrastructure, such as a cellular environment or a satellite. One of the goals we are currently investigating is how to quickly set up routing tables in the nodes with the help of these two communication modes so that messages can be sent along the near-shortest paths in the local network. Solutions to this problem could be used in modern smartphones in order to significantly improve the efficiency of direct interactions between these. Another interesting direction is cloud-assisted overlay networks. The cloud offers a highly scalable solution to classical client-server-based approaches. However, the cloud is not for free. Thus, it would be desirable to use the cloud only to assist an overlay network, instead of taking over its role. Finding the right balance between using the cloud and an overlay network in order to come up with highly scalable and robust solutions for certain applications appears to be a challenging problem.

In our research on heterogeneous resource allocation problems, we explored various extensions. For example, we presented results on uniform metrics for the heterogeneous k-server problem, but further study is needed for more complex metrics. Additionally, considering more complex request types, such as subsets of servers, can significantly increase the problem's difficulty. Similar extensions can be applied to other resource allocation problems like multi-commodity facility location. Our research aims to connect the influence of heterogeneity with classical resource allocation through parameterized competitive ratios. As an example of general ways to step away from a worst-case measure, recently, models became popular where the online algorithm can utilize the advice of a machine learning algorithm to make better choices. Here, the difficulty lies in obtaining improvements when the advice is good while keeping the worst-case guarantees of classical online analysis when the advice is bad. Such techniques allow the research on online algorithms in general to approach the often far better performance observed in practice.

Bibliography

[AAG09]   ALON, N.; AZAR, Y.; GUTNER, S.: Admission control to minimize rejections and online set cover with repetitions. In: *ACM Transactions on Algorithms* 6 (2009), no. 1, 11:1–11:13

[ACD+11]   ABSHOFF, S.; CORD-LANDWEHR, A.; DEGENER, B.; KEMPKES, B.; PIETRZYK, P.: Local Approx-
            imation Algorithms for the Uncapacitated Metric Facility Location Problem in Power-
            Aware Sensor Networks. In: *Algorithms for Sensor Systems - Proceedings of the 7th
            International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks
            and Autonomous Mobile Entities (ALGOSENSORS)*. Vol. 7111. 2011, pp. 13–27

[AG07]   ANTHONY, B. M.; GUPTA, A.: Infrastructure Leasing Problems. In: *Proceedings of the
            12th International Conference on Integer Programming and Combinatorial Optimization
            (IPCO)*. Vol. 4513. 2007, pp. 424–438

[AHK+20]   AUGUSTINE, J.; HINNENTHAL, K.; KUHN, F.; SCHEIDELER, C.; SCHNEIDER, P.: Shortest Paths in a
            Hybrid Network Model. In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete
            Algorithms (SODA)*. 2020, pp. 1280–1299

[AKM+16]   ABSHOFF, S.; KLING, P.; MARKARIAN, C.; MEYER AUF DER HEIDE, F.; PIETRZYK, P.: Towards
            the price of leasing online. In: *Journal of Combinatorial Optimization* 32 (2016), no. 4,
            pp. 1197–1216

[AMM14]   ABSHOFF, S.; MARKARIAN, C.; MEYER AUF DER HEIDE, F.: Randomized Online Algorithms
            for Set Cover Leasing Problems. In: *Proceedings of the 8th International Conference on
            Combinatorial Optimization and Applications (COCOA)*. Vol. 8881. 2014, pp. 25–34

[BS89]   BLACK, D.; SLEATOR, D.: *Competitive Algorithms for Replication and Migration Problems*.
            Technical Report CMU-CS-89-201. Department of Computer Science, Carnegie-Mellon
            University, Jan. 1989

[CCF+22]   COY, S.; CZUMAJ, A.; FELDMANN, M.; HINNENTHAL, K.; KUHN, F.; SCHEIDELER, C.; SCHNEIDER,
            P.; STRUIJS, M.: Near-Shortest Path Routing in Hybrid Communication Networks. In: 217
            (2022), 11:1–11:23

[CCS+23]   COY, S.; CZUMAJ, A.; SCHEIDELER, C.; SCHNEIDER, P.; WERTHMANN, J.: *Routing Schemes for
            Hybrid Communication Networks*. 2023

[CFK+20]   CASTENOW, J.; FELDKORD, B.; KNOLLMANN, T.; MALATYALI, M.; MEYER AUF DER HEIDE, F.:
            The Online Multi-Commodity Facility Location Problem. In: *Proceedings of the 32nd
            ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. July 2020,
            pp. 129–139

[CFK+22]   CASTENOW, J.; FELDKORD, B.; KNOLLMANN, T.; MALATYALI, M.; MEYER AUF DER HEIDE, F.:
            The K-Server with Preferences Problem. In: *Proceedings of the 34th ACM Symposium on
            Parallelism in Algorithms and Architectures (SPAA)*. July 2022, pp. 345–356

[CLP21]   CENSOR-HILLEL, K.; LEITERSDORF, D.; POLOSUKHIN, V.: Distance Computations in the Hy-
            brid Network Model via Oracle Simulations. In: *Proceedings of the 38th International
            Symposium on Theoretical Aspects of Computer Science (STACS)*. Vol. 187. 2021, 21:1–
            21:19

[Dij74]   DIJKSTRA, E. W.: Self-Stabilizing Systems in Spite of Distributed Control. In: *Communica-
            tions of the ACM* 17 (Nov. 1974), no. 11, pp. 643–644

[FHS20]   FELDMANN, M.; HINNENTHAL, K.; SCHEIDELER, C.: Fast Hybrid Network Algorithms for
            Shortest Paths in Sparse Graphs. In: *Proceedings of the 24th International Conference on
            Principles of Distributed Systems (OPODIS)*. Vol. 184. 2020, 31:1–31:16

[FKM22]   FELDKORD, B.; KNOLLMANN, T.; MEYER AUF DER HEIDE, F.: Online facility location with
            mobile facilities. In: *Theory of Computer Science* 907 (2022), pp. 45–61

[FKMM21]   FELDKORD, B.; KNOLLMANN, T.; MALATYALI, M.; MEYER AUF DER HEIDE, F.: Managing Multiple
            Mobile Resources. In: *Theory of Computing Systems* 65 (2021), no. 6, pp. 943–984

[FM19]   FELDKORD, B.; MEYER AUF DER HEIDE, F.: The Mobile Server Problem. In: *ACM Transactions
            on Parallel Computing* 6 (2019), no. 3, 14:1–14:17

[Fot08]   FOTAKIS, D.: On the Competitive Ratio for Online Facility Location. In: *Algorithmica* 50
            (2008), no. 1, pp. 1–57

[FS19]   FELDMANN, M.; SCHEIDELER, C.: Skeap & Seap: Scalable Distributed Priority Queues for Constant and Arbitrary Priorities. In: *Proceedings of the 31st ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2019, pp. 287–296

[FSS18a]   FELDMANN, M.; SCHEIDELER, C.; SETZER, A.: Skueue: A Scalable and Sequentially Consistent Distributed Queue. In: *Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2018, pp. 1040–1049

[FSS18b]   FELDMANN, M.; SCHEIDELER, C.; SETZER, A.: Skueue: A Scalable and Sequentially Consistent Distributed Queue. In: *CoRR* abs/1802.07504 (2018). arXiv: `1802.07504`.

[FSS21]   FELDMANN, M.; SCHEIDELER, C.; SCHMID, S.: Survey on Algorithms for Self-stabilizing Overlay Networks. In: *ACM Computing Surveys* 53 (2021), no. 4, 74:1–74:24

[KMP12]   KLING, P.; MEYER AUF DER HEIDE, F.; PIETRZYK, P.: An Algorithm for Online Facility Leasing. In: *Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*. Vol. 7355. 2012, pp. 61–72

[KS22]   KUHN, F.; SCHNEIDER, P.: Routing Schemes and Distance Oracles in the Hybrid Model. In: *Proceedings of the 36th International Symposium on Distributed Computing (DISC)*. Vol. 246. 2022, 28:1–28:22

[Mey01]   MEYERSON, A.: Online Facility Location. In: *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2001, pp. 426–431

[Mey05]   MEYERSON, A.: The Parking Permit Problem. In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2005, pp. 274–284

[MMS88]   MANASSE, M. S.; MCGEOCH, L. A.; SLEATOR, D. D.: Competitive Algorithms for On-line Problems. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. 1988, pp. 322–333

[NW13]   NAGARAJAN, C.; WILLIAMSON, D. P.: Offline and online facility leasing. In: *Discrete Optimization* 10 (2013), no. 4, pp. 361–370

[RS04]   RAVI, R.; SINHA, A.: Multicommodity facility location. In: *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2004, pp. 342–349.

[SSS15]   SCHEIDELER, C.; SETZER, A.; STROTHMANN, T.: Towards Establishing Monotonic Searchability in Self-Stabilizing Data Structures. In: *Proceedings of the 19th International Conference on Principles of Distributed Systems (OPODIS)*. Vol. 46. 2015, 24:1–24:17

[SSS16]   SCHEIDELER, C.; SETZER, A.; STROTHMANN, T.: Towards a Universal Approach for Monotonic Searchability in Self-stabilizing Overlay Networks. In: *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*. Vol. 9888. 2016, pp. 71–84

[ST85]   SLEATOR, D. D.; TARJAN, R. E.: Amortized Efficiency of List Update and Paging Rules. In: *Communications of the ACM* 28 (1985), no. 2, pp. 202–208

[STA97]   SHMOYS, D. B.; TARDOS, É.; AARDAL, K.: Approximation Algorithms for Facility Location Problems (Extended Abstract). In: *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*. 1997, pp. 265–274

Subproject A3:

The Market for Services: Incentives, Algorithms, Implementation

Claus-Jochen Haake[1], Burkhard Hehenkamp[1], Gleb Polevoy[2]

1 Department of Economics, Paderborn University,
  Paderborn, Germany
2 Heinz Nixdorf Institute and Department of Computer
  Science, Paderborn University, Paderborn, Germany

1   Introduction

The arguably widest definition of a **market** is the *organization of interaction across several agents*. The term *interaction* can thereby be filled in different ways, meaning that the market participants face various strategic possibilities to take action. The goal of Subproject A3 "The Market for Services: Incentives, Algorithms, Implementation" is to take a close look at interactions within and between the different groups of agents in an OTF market with the aim to elicit incentive structures, to evaluate the outcomes, or to design rules according to which the agents are supposed to act. Analyzing given interaction forms either means identifying equilibria, which enables us to predict the behavior of the market participants and thus gives us a tool to assess the consequences on individual and social welfare.
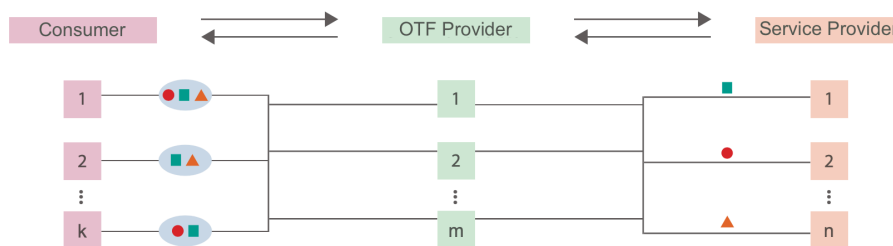
Figure 3: *Layout of an OTF market.*

The general layout of an OTF market is illustrated in Figure 3, which shows a number of specific characteristics. First, we may (ideally) view it as a two-sided market with consumers (customers, end users) one one side and service providers on the other side. OTF (service) providers act as mediators between consumers and service providers. However, unlike in traditional models for two-sided markets, OTF service providers do more than just mediate. Based on a customer's request, they actively form service compositions for which they act as a demander of services themselves. In relation to the customer, an OTF service provider takes the role of a seller to satisfy th customer's demand. As a result,

---
cjhaake@wiwi.upb.de (Claus-Jochen Haake), burkhard.hehenkamp@upb.de (Burkhard Hehenkamp),
gpolevoy@mail.upb.de (Gleb Polevoy)

contracts are not only signed between the customer and the OTF provider, but also between the OTF service provider and possibly several service providers.

Within this market structure we investigate several different forms of interaction using methods from microeconomics as well as from non-cooperative, cooperative, and algorithmic game theory. Interaction takes place on different levels, i.e., involving smaller or larger groups. Identifying the agents' incentives and the design of institutions that lead them in the right direction is at the heart of Subproject A3.

As an example, we consider the contracting problem between a single OTF service provider and a service provider. On this micro-level, the question is how to find a fair and efficient way to settle negotiations on the terms of trade. The models we use here stem from cooperative game theory or, more specially, from bargaining theory. It turns out that there is a tradeoff between fairness and efficiency of the outcome. While the former is attractive to market participants, the latter property is, given an appropriate business model, interesting to the OTF market provider.

Taking a slightly broader perspective, an antecedent problem occurs: namely, who contracts with whom in the market. Such problems are central in matching theory, which designs and studies mechanisms that match agents from two different sides (in a market) in a preferably efficient and stable manner. Especially when capacity constraints (of OTF providers) play a role, the answer to the question of which end users should be matched to which OTF service providers could have a crucial impact on market performance. Further, matching mechanisms themselves provide incentives to act strategically and thus have to be analyzed and controlled in this direction.

From an overall view, competition naturally is a vital issue and raises a couple of questions such as how prices are formed or how the structure of offered services evolves. Concerning the latter, there may arise incentives to offer particular services only in combination with other services (bundling). From a mechanism design point of view, welfare analyses are inevitable to find directions in which the market should be regulated.

Finally, the success of an OTF market is ultimately linked to information on services that is as clean as possible. While from the outset, incomplete information on service qualities is a typical characteristic of an OTF market, the design of information systems that are based on user ratings help to reduce the lack of information, reduce incentives for misbehavior and hence increase efficiency of the market outcome. Again, on the one hand it is our task to investigate the participants' incentives to react to a well-functioning rating system that aims at distinguishing high and low quality services. On the other hand, the challenge is to elicit and process information on observed product qualities so that unwanted behavior can readily be detected.

In the next section, we highlight contributions from Subproject A3 concentrating on economic issues of the market. Abstracting from technical aspects of service composition, service execution, the organization of the infrastructure, or security issues, our task is to analyze the behavior of market participants from a mostly theoretical perspective, including the provision oof methods for efficient computation of solutions.

2   Highlights and Lessons Learned

This section is structured in three subsections highlighting our results. Subsection 2.1 focuses on specific scenarios in which (OTF) providers compete with each other and investigates the effects on welfare and market structure. In Subsection 2.2 we are concerned with a more individual level of interaction and implementation of outcomes. Further, we take a closer look at how quality standards can be maintained through reputation systems. Finally, Subsection 2.3 highlights our results on algorithms developed to calculate and hence predict market outcomes such as equilibria.

2.1   Competition

We take four different perspectives in our analysis of competition in OTF markets. First, we investigate the incentives to combine elementary products to compositions of products rather than to sell them as separate entities.[1] Second, we study experimentally to what extent implementing competition can help improve the quality of services in (OTF) markets that are characterized by fixed or regulated prices. Third, we explore whether opening a monopolistic (OTF) market for competition improves market performance by increasing the quality of products when the market entrant has the option to differentiate its product away from the already existing one. Fourth and finally, we examine how competition for innovation is affected when firms struggle for their survival in the market.

Bundling

In our first highlight, we explore a question that lies at the heart of any OTF market [EHH22]. Under which conditions is the composition of products, i.e., *bundling*, optimal to an OTF provider at all? To what extent does the incentive to sell products in bundles rather than in separate entities depend on the nature of the original elementary products such as their degree of product differentiation. Does the incentive depend on whether the elementary products represent substitutes or complements? Does it depend on their respective degrees of substitutability or complementarity?

To investigate these issues we use a rather specific asymmetric market setup, which we describe in detail further below. Additionally, we show that this specific market setup may arise endogenously in a richer setup, where service providers are enabled to choose their distribution channels optimally. Finally, we also take a social perspective and examine the welfare consequences of bundling in our framework.

Figure 4 below illustrates our market setup. We consider two retailers R_A and R_B and two monopolistic service providers M_1 and M_2, the latter of which each produce a differentiated elementary product. The retailers compete in prices. Service provider M_2 sells its product to both retailers, while M_1 only supplies retailer R_A. Retailer R_A hence receives both elementary products and considers whether or not to sell them as a bundle or as separate entities.

Accordingly, retailer R_A can be viewed as a firm deciding whether or not to adopt the role of an OTF provider. To examine retailer R_A's incentive for bundling, we determine how

---

[1]While market participants may compete in products or services (or both) in most of our projects, we shall henceforth write either 'products' or 'services' implicitly including the other meaning(s) as well.
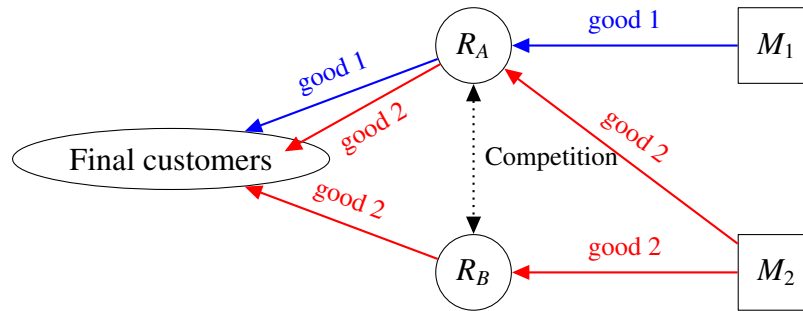
Figure 4: *The market setup.*

the degree of product differentiation of the elementary products affects the equilibrium prices, quantities and profits for each of the two selling alternatives: separate selling and bundling.

Our main result is the following: Retailer R_A will only find it optimal to bundle the elementary products when they represent close substitutes. This might sound counterintuitive at first, but we can frequently observe this form of bundling in the real world. For instance, grocery stores sell packs of peppers in bundles, either as almost perfect substitutes of identical color or as close substitutes in different colors. Similarly, clothing shops commonly offer bundles of socks or pants that only vary in patterns or colors from each other.

The intuition of our first result runs as follows. When the elementary products are differentiated, bundling alleviates competition in two ways. On the one hand, it eliminates the perfect substitutability of the elementary product 2, which is available from both retailers. On the other hand, bundling also creates a complementarity between the elementary products 1 and 2. Competition between differentiated products is the stronger the lower their degree of differentiation. Consequently, the anticompetitive effect of bundling is strongest when the elementary products constitute close substitutes, since then competition is intense. Only in this case, is the competition-reducing effect of bundling so strong that it outweighs the aggravation of the double marginalization problem that occurs along the vertical supply chain. Therefore, bundling by retailer R_A in our market setup is only profitable when the elementary products represent close substitutes. Only then might retailer R_A actually adopt the role of an OTF provider.

With regard to social welfare, the picture is blurred. Product bundling reduces the consumer surplus because of the higher downstream prices, while it increases the producer surplus, since all firms earn a higher profit, both in the downstream and the upstream market. Social welfare, however, only increases through bundling if the elementary products constitute close to perfect substitutes.[2] Thus, the emergence of OTF providers must be viewed quite critically from a social perspective.

Finally, we extend our framework to incorporate the service providers' choice of their distribution channels. As it turns out, our asymmetric market setup indeed represents an equilibrium outcome of this extended model when the elementary products represent close but not too close substitutes. Moreover, in this case, bundling reduces social welfare and should be prohibited.

---

[2]Recall that social welfare is defined as the sum of consumer surplus and producer surplus.

To sum up, our first highlight stresses the importance of product differentiation for bundling elementary products when the associated vertical market structure is prone to double marginalization. At least within our market framework, product bundling, and hence OTF providers, should raise serious antitrust concerns.

Introducing Competition I

Many healthcare markets can be viewed as instances of two-sided OTF markets with indirect network externalities. For example, hospitals match patients with various types of diseases with doctors from various fields of specialization. The more fields of specialization a hospital covers, the more likely it is that a patient will find the appropriate treatment. The more patients there are, the more likely it is that a doctor of a certain specialization will find patients.[3] Similarly, patients attending a general practitioner (GP) benefit from the experience the GP has gained from treating other patients. The more other patients there are, the better the experience of the GP.

This second highlight and the third one further below both examine monopolistic provider markets that are opened for competition. The aim is to evaluate whether opening the market for competition improves the market outcome in terms of quality, customer benefit or social welfare. Furthermore, we study the consequences of the providers' customer orientation (such as physician altruism) for the market outcome. In this second highlight, our focus is on the quality of the service, while the type of service is taken as given. Moreover, we deploy an experimental approach to study the effect of introducing competition. In contrast, the third highlight further below scrutinizes a theoretical model, where the new provider chooses its quality of service but may also adjust the type of service offered in order to soften competition.

Previous research has shown that, without competition, providers deviate from the customer-optimal provision under payment systems such as capitation and fee-for-service. While capitation corresponds to a fixed payment per treatment, the total payment under fee-for-service depends on the number of services executed within a given treatment. Correspondingly, a profit-maximizing provider would execute no services at all under capitation, while it would perform the maximum number of feasible services under fee-for-service. While competition is expected to mitigate these distortions, providers usually interact with each other repeatedly over time and only a fraction of customers switches providers at all. Both features might prevent the desired effect from introducing competition.

We consider two setups ([BHK17], [BHK23]). In both setups, we experimentally study the effect of introducing competition among providers when there is a trade-off between the choice of maximizing customer utility and the choice of maximizing a provider's profit. While in [BHK17] providers face homogenous customers that all face a problem of identical severity, [BHK23] scrutinizes the effects that originate from a heterogenous customer population. For both setups, we develop a theoretical model that serves as our benchmark, which we then test in a controlled laboratory experiment.

In [BHK17], our experimental conditions vary the physician payment scheme (capitation vs. fee-for-service) and the severity of the patient's problem (high vs. low). Real patients benefit from the provider decisions made in the experiment. We find that, in line with

---

[3]Notice that there might be also indirect network externalities, which are negative. For instance, the number of nurses per patient is decreasing in the number of patients treated.

the theoretical prediction, introducing competition can reduce underprovision (under capitation) and overprovision (under fee-for-service). The strength of the observed effects, however, depends on the severity of the problem and the payment scheme. We also find providers to collude tacitly, in particular under fee-for-service payment. Collusion appears less often than in related experiments on price competition though.

In [BHK23], providers face heterogenous customers that differ in the severity of their problem and in their mobility. Mobile customers choose their provider on the basis of the (expected) benefit from treatment, while immobile patients always visit the same provider. While we also examine the effect of introducing competition, the analysis of the second setup centers on the effects of customer heterogeneity on the market outcome. In line with the theoretical prediction, we find that introducing competition significantly increases patient benefit for mobile patients. In contrast, for immobile patients, competition worsens the outcome compared to a situation without competition. This latter observation does not match with our theoretical prediction, which would predict no difference. With repetition of the interaction both effects become more pronounced. Our results imply that introducing competition does not entail unique positive effects, but rather ambiguous effects that differ across customer groups. In particular, customer mobility is decisive for the market outcome.

Introducing Competition II
In the second highlight, we evaluate whether opening a formerly public (or private) price-regulated monopoly market for competition represents a viable option for improving quality and choice for customers. To this end, the welfare effects from opening the market are determined.

In price-regulated monopoly markets we typically observe low product quality. In principle, opening the market for competition could be a good idea if the entrant(s) offered the same or a similar product. Then, quality competition would be intensified and firms would offer higher levels of quality to attract demand. On the other hand, entrants face an incentive to avoid or at least soften competition by offering a product that differs sufficiently from the original one. As a consequence, quality might not increase to the extent expected in the first place.

To explore our research question, we consider a three-stage duopoly model of location choice and quality competition with price regulation and costly relocation. There are three active players: a budget-constrained regulator, the incumbent monopolist, and the entrant. At stage 1, the regulator sets the price. At stage 2, the entrant chooses a location, while the incumbent monopolist is already located at the center of the market and it is too costly for him to relocate. At stage 3, the two firms compete in quality for customers' demand. Observe that the location choice of the entrant at stage 2 exhibits the trade-off between moving away from the incumbent to soften competition (the so-called *competition effect*) and moving closer to steal demand (the *demand effect*).

We consider two setups, both of which are relevant in markets such as public health care, education and schooling, or postal services ([HK20], [HK23]). In [HK20], the incumbent monopolist represents a public provider, e.g., a hospital, with some degree of customer-orientation, for example, since the hospital's physicians show some degree of altruism towards their patients. In addition, the public provider faces a profit constraint that prohibits

losses. The entrant, in contrast, maximizes pure profit. In [HK23], both the incumbent monopolist and the entrant are for-profit providers, each one maximizing its own profit.

Our main results are as follows: In [HK20], opening a public hospital market typically raises quality. The private provider strategically locates towards the corner of the market to avoid (too) costly quality competition. The consequences for social welfare depend on the size of the regulator's budget and on the degree of customer orientation of the public provider. If the regulator's budget is large, high quality is implemented and welfare is highest in a duopoly whenever entry is optimal. However, when the budget is small, quality levels in the duopoly correspond to the monopoly level. In particular, it turns out that for intermediate budgets it can be optimal for the regulator to not use the entire budget.

In [HK23], the entrant strategically locates towards the corner of the market, keeping the incumbent at the monopoly quality level when the regulated price is low or intermediate. In this case, quality is only raised for the entrant's customers. When the price is high, the entrant locates at the corner of the market and both providers implement a higher quality compared to the monopoly level. Moreover, the entrant always implements a higher quality than the incumbent provider. Social welfare is always higher in a duopoly if the cost of quality is low. For higher levels of quality cost, welfare is non-monotonic in the price. Therefore, the regulator will optimaly withhold part of its budget for certain budget sizes. Finally, the welfare effect from opening the market for competition depends on the price and the size of the entry cost and the decision to allow entry should be conditioned on an assessment of the entry cost.

Firm survival and innovation
Taking a dynamic perspective on competition, not only the existing products matter for the outcomes of competition in a market but also the incentives to come up with new products, new services or new technologies. This is where our last highlight departs [GHL19]. We want to explore the outcomes of competition in innovation contests where a finite number of firms potentially compete with each other for an innovation. In particular, we want to investigate the case in which the number of firms actually competing for the innovation is uncertain and in which the behavior of a firm is governed by the strive for survival of the firms.

Competition for innovation can assume different forms. For instance, there are prizes announced for certain ideas or solutions, e.g., for algorithms that manage to accomplish a certain task, or there always is the option to register an innovation as a patent. The monopoly right that comes with a patent can then subsequently be exploited by marketing the innovation as a new product or service.

Competition for innovation is characterized by three unique features, the combination of which distinguishes it from competition in product markets. First and foremost, competition for innovation is dynamic, which makes it necessary to incorporate a time dimension in the analysis. Second, the success of investments in innovation is highly uncertain and depends on the investments made and the outcomes realized by a firm's competitors. Moreover, not always the firm with the highest investment will succeed in winning the patent. Third and finally, the investments of *all* competitors are sunk, also those made by the unsuccessful firms. Accordingly, a dynamic contest model with imperfect discrimination represents the appropriate model to study our questions at hand.

To model uncertainty about the number of competitors, we investigate two setups. In the first, we consider stochastic participation, that is, a single firm does not know the number of competitors that also participate in the same contest. From the perspective of the firm, it appears *as if* other firms participate in the contest with a certain probability. In addition, the number of potential competitors in the contest is exogenously given. In the second setup, we depart from the latter assumption to determine the number of potential competitors endogenously. To this end, we introduce a fixed cost of entry.

Markets with a high innovation intensity tend to be very dynamic and subject to high fluctuations. The entry and exit of firms are the rule rather than the exception. Correspondingly, the behavior of a firm is better described as governed by the firm's striving for survival than by profit maximization. To account for this, we deploy the so-called economic evolutionary approach.

Under the economic evolutionary approach, the biological evolutionary forces of selection, mutation, and heredity correspond to economic evolutionary forces such as imitation, innovation, and bankruptcy to name but a few. (Economic) evolutionary equilibrium then serves as the short-cut to the evolutionary outcome of a dynamic evolutionary process of imitation and innovation. Finally, we consider finite (firm) populations as the economically relevant case and solve our model for the corresponding equilibrium of a finite-population evolutionarily stable strategy.

Apart from solving for the economic evolutionary equilibrium, our focus is on the *issue of (over-)dissipation.* This issue is closely related to Posner's (1975) famous full dissipation hypothesis, according to which competition for a monopoly (in our case: a patent monopoly) would eat up the entire monopoly rent that the firms compete for. In our paper, we re-evaluate Posner's hypothesis. In our setup of a finite population, the strive for survival leads to more aggressive investment behavior, so the issue might be particularly pronounced.

Our main findings are as follows. Firstly, when the probabilities of participation are exogenously given, competitors choose higher levels of investment in the economic evolutionary equilibrium than in the Nash equilibrium. Moreover, there is ex-ante overdissipation in the economic evolutionary equilibrium for sufficiently large probabilities of participation if, and only if, the impact function is convex.[4] These results generalize earlier findings in [HLP04] from contests with a given, i.e. deterministic number of firms to contests with stochastic participation, where the number of actual competitors is *a priori* uncertain.

Secondly, with costly endogenous entry, firms enter the contest with a higher probability and choose higher levels of investment in the economic evolutionary equilibrium than in the Nash equilibrium. Importantly, under endogenous entry, overdissipation can occur for all types of contest technologies, in particular those with concave impact functions.

Our findings point to potentially high welfare losses stemming from innovation contests when they are open to anyone.

---

[4] An *impact function* can be understood as a lottery production function, which governs the transformation of investments into probabilities of winning. A *convex / linear / concave* impact function then exhibits increasing / constant / decreasing returns to investments.

2.2 Allocation and Incentives

In the OTF market interaction takes place in various environments. Especially when there are few participants, competition may not be a plausible form of modeling market organization. Therefore, we analyze alternative models of interaction such as bargaining, matching or, more generally, mechanism design. Because there is naturally no complete information on service qualities, we need to keep track of the agents' incentives to misbehave and exploit a superior information position at the cost of efficiency. In this subsection we review some of our results in the above fields.

Bargaining

The term *bargaining* generally refers to a situation, in which two or more persons can sign an agreement on the distribution of joint gains. A *bargaining solution* is supposed to propose such an agreement for any possible bargaining problem (from a specific set of problems), i.e., not focusing on a particular problem. The main challenge is to define or design bargaining solutions that capture the context of interaction and in which agreements are to be considered as *fair*. In this section, we briefly review two works that argue for the selection of particular bargaining solutions in bargaining problems occurring on an OTF market.

Some particular interactions in the OTF market involve two participants only. For instance, consider the situation, in which an OTF provider and a service provider negotiate over the terms of trade, including the price. Since we consider highly specialized services, there is typically no market, in which a price can be settled by an equilibrium mechanism, which means it is subject to bilateral negotiations. To aggravate the problem, both parties may have private information on either production costs or expected revenues from sales to the customer. In a simplified version, the OTF provider and the OTF service provider negotiate over a service level (e.g., quantity, quality degree, etc.) and a total payment. The presence of incomplete information requires the two parties bargain over contracts that are type dependent, i.e., those that depend on the realization of costs and revenues that have to be reported by the parties. Because the final agreement (payment, quantity, etc.) relies on the report of unobservable private information, it is naturally open to cheating. As a result, a "good" contract should satisfy a number of properties such as (i) ex post Pareto efficiency (EPE), (ii) individual rationality (IR), and (iii) incentive compatibility (IC). While EPE ensures that there is no room for renegotiations after the contract has been agreed upon, IR provides incentives to enter negotiations at all. Finally, IC requires that truthful reporting be a Nash equilibrium in the reporting game.

A closer inspection of the structure of the negotiation problem reveals that it exhibits the features of an intra-firm transfer pricing problem. A transfer pricing problem as it is studied in the literature involves two divisions of the same company and is displayed in Figure 5. The aim is to settle an agreement on how much and at what price an intermediary good is internally sold from the producing division to the buying division. While the producing division has better information on production costs, the sales division knows the external market on which the product is finally sold to customers. Besides the goals of the divisions that act as profit centers, the central management prefers a company-optimal outcome. Returning to our negotiation in the OTF market, the additional quantity parameter may be interpreted as such if a hardware service is under concern. Alternative interpretations may

include the duration of service provision. The role of the company could be played by an OTF market maker who is interested in an efficient market outcome.
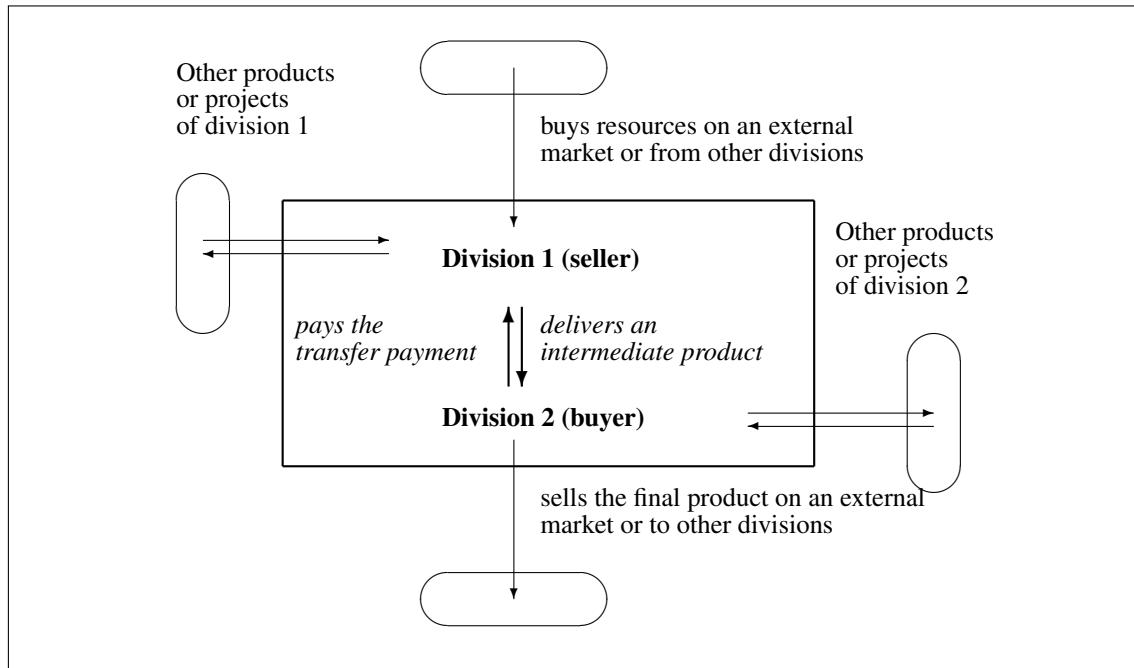
Figure 5: *Intra-firm Transfer Pricing Problem. The two divisions bargain over a payment from division 2 to division 1 in return for a quantity of the product. (Source: [HR18]).*

In [HR18] we propose a fair solution to the transfer pricing problem that rests on a solid foundation in bargaining theory and which is new to the transfer pricing literature. We mainly use the transfer pricing scenario as we compare the solution to other well-known solutions from this field. To be more precise, for a transfer pricing game under incomplete information we determine the generalized Nash bargaining solution. Requiring agreements to be incentive compatible and/or efficient, we further highlight the relation between these two desirable properties. For a necessary intermediate result for the applicability of the generalized Nash bargaining solution, we show that the transfer pricing game is regular, meaning that it is possible to guarantee each division a strictly positive expected profit, regardless of their specific private information. From a managerial perspective, the appealing feature of the generalized Nash bargaining solution is that it provides each division with a strictly positive expected profit. Further, we derive necessary conditions for a mechanism that implements the generalized Nash bargaining solution (Propositions 4, 5, and 6) and shed light on the trade-off between efficiency and fairness (Proposition 7). As illustrated in examples, the Nash solution tends to keep differences in divisional profits smaller in comparison to other solutions. Two examples illustrate differences between the generalized Nash bargaining solution and well-established alternatives from [Wag94].

In sum, we find that if parties are interested in a fair outcome, our analysis provides good arguments to use the generalized Nash bargaining solution. For the bargaining problem on an OTF market, we may think of bargaining problems as being automatically resolved in a way that takes fairness and efficiency into account. While the former increases the attractiveness of an OTF platform, the latter increases the incentives of a market maker to

provide the platform itself.

The nature of a (cooperative) bargaining problem is to distribute common gains in a fair way. Apart from bargaining particular bargaining situations in which parties bargain once, we also investigated problems of repeated interaction from a structural point of view. [Hoo20] analyzes a model in which the bargaining problem is shaped by the possible payoffs from strategies in a differential game, i.e., a non-cooperative game that is played in continuous time. Thus, in this scenario the participants may share common gains over time. In terms of fairness, this requires that solutions be individually rational over time and consistent with time preferences, which can be thought of as discount factors for future payoffs. The main result is that for a class of underlying differential games, both properties are already fulfilled when the bargaining solution satisfies an overall individual rationality. One advantage of the latter property is that it is intuitive, as it guarantees participation in the interaction and in accordance with the main result, it triggers consistency over time. From a theoretical point of view, this has an impact on which bargaining solution should or should not be selected by a market designer.

Matching

The problem of finding a good market allocation is directly connected to the question, who serves whom in the market. There is a still growing literature on two-sided matching markets that discusses algorithms for matching agents from one group to agents of another group. The assessment whether an algorithm is "good" or "bad" is ultimately linked to the properties of the final outcome. Besides the efficiency, stability of the matching plays the most important role. It guarantees that no agent or group of agents would want to alter the matching and have the possibilities to do so. Examples for classical matching markets are the marriage market and college admissions [GS62], school choice [AS03] and the housing market [SS74].

In an OTF market, specific allocation problems can be viewed as a matching market – end users have to be matched to OTF service providers, while service providers are matched to OTF service providers. The ingredients of a matching market are the participants' preferences over participants on the other side. For example, differences in the characteristics of an OTF provider or the heterogeneity of traded composed services form a consumer's preferences over providers, while the users' different demands or their willingness to pay shape an OTF service provider's preferences over users. Regarding an OTF market as being organized on a (central) platform operated by an OTF market provider, the matching problems can be described as a many-to-one matching market, which in the literature is commonly termed the *college admission problem (CAP)* or school choice problem.

We address three main problems connected to desirable matchings in the OTF market: (1) How can matching algorithms be adjusted to cope with users' heterogeneous demands? (2) Since the market is large, a participant may not know all options on the other side, but still has to form preferences. Therefore, how can incomplete information on the participants be dealt with? (3) Finally, how is the functioning of matching mechanisms affected, when the formation of preferences follows an (automatic) pattern?

Question 1 addresses a problem that is widely ignored in classical matching models. In a CAP, students are matched to exactly one seat at a college, so that all have the same weight or need of capacity. The total number of available seats, or total capacity, belongs to a

college's characteristics ([GS62; AS03]). For this scenario, the Boston Mechanism (BM), the Deferred Acceptance Algorithm (DA), and the Top Trading Cycle Algorithm (TTC) are the most used mechanisms in practice. But if the homogeneous demand assumption that each student requires exactly one seat is dropped, only little is known. In weighted matching markets or matching markets with sizes [BM14], stability can no longer be assured [MM10].

In [HS20b] we start from the fact that stability is no longer assured and investigate how we can find a stable outcome, if possible, and how to enable it otherwise. For this, we introduce a new algorithm, the deferred acceptance algorithm with gaps, which either results in a stable matching, if one exists, or leads to a cycle. If all students have the same weight, meaning that they all need the same amount of capacity, the algorithm operates exactly as the deferred acceptance algorithm [GS62]. However, if the algorithm leads to a cycle, because there is no stable matching, we can arrive at stability by increasing or decreasing the colleges' capacities.

As stability is no longer guaranteed unless we modify capacities, it is also possible to have a look at Pareto efficient outcomes of weighted matching problems or, more precisely, weighted school choice problems. [Str20] proposes a variant of the TTC algorithm: namely, the weighted TTC (WTTC), which is strategy-proof and yields a Pareto efficient outcome. But although the main results carry over compared to the TTC, the usage of the WTTC introduces a trade-off between weights and preferences or priorities. Thus, the introduction of weights comes with some costs as it is more complex to guarantee each student a seat at a college.

Addressing the second question, one source of incomplete information over the other participants' preferences comes from the bare size of the OTF market. Incomplete information was introduced by [Rot89]. Given a strategy-proof matching algorithm, he shows that truth-telling is still a dominant strategy if agents only have limited information about the other agents' preferences. While there is already some experimental research on the functioning of algorithms that are not strategy-proof in a setting of incomplete information [CS06; CLS16], no empirical evidence exists on the behavior of students in a school choice problem with incomplete information when a variant of the Boston school choice mechanism is used. The BM does not yield a stable or Pareto efficient outcome and, most importantly, is not strategy-proof [AS03].

[HS20a] fills this lack and further introduces heterogeneity in the market by taking into account different weights of students. Our research uses two different data sources, the data derived from the clearinghouse that implements the matching algorithm itself and data from a voluntary survey among the students who participated in the clearinghouse. We find that over 74% of students misrepresent at least one of their ranks in the preference list, which is not surprising given that the algorithm is not strategy-proof. But although students are trying to exploit incentives, they do not necessarily succeed in improving their outcomes through manipulation. This is mainly due to the fact that students have incomplete information on the other students' preferences. Additionally, some students do better in misrepresenting than others. We call these students sophisticated, whereas another group of students is not able to act in a consistent manner and is thus naive. This notion of sophistication is based on the theoretical definition by [PS08]. We see that sophisticated students actually reach significantly better outcomes than naive students.

Another source of incomplete information may arise on the other market side, which brings us to the third question. In school choice it is assumed that schools are not able to rank all the students individually but with the help of some objective criteria [AS03]. But if these criteria are based on the students' preferences this might actually undermine the functioning of any matching mechanism. More precisely, [HS18] analyzes what happens if the schools' priorities are formed in a reciprocal way, i.e., based on the students' preferences in a "first-preference-first" manner. We show that in this case the deferred acceptance algorithm, the TTC, and the Boston school choice mechanism all yield the same outcome and are thus manipulable. This means that even in otherwise strategy-proof mechanisms, it is no longer a dominant strategy to state the true preferences.

To sum up, our works provide insights on how matching processes shall be set organized in an OTF market. There, heterogeneity of agents and incomplete information on the matching problems. The results show that special care has to be taken on the design strategy proof algorithms and on the formation of preferences to exclude unwanted behavior or the emergence of misdirected incentives.

Quality assurance: Customer evaluations

Addressing the question of how a good or poor service quality can be identified, we review two papers on how customer feedback can be of assistance. One of the characteristics of almost any market and in particular of OTF markets is that information on service qualities is asymmetrically distributed. One reason for this is that services are typically experience goods. Such goods do not reveal their true qualities to the consumer prior to purchase and consumption. In that spirit, whether a particular service from a service provider or the service composition sold by the OTF service provider actually delivers the desired result to the end user can only be verified after the transaction between user and OTF service provider has taken place.

A particular consequence of the observability of service quality after purchase is that it opens the door for strategic interaction on the provider's side. Because lower service quality typically comes at lower costs, an optimal decision would be not to produce high quality services. To make the problem even more demanding, service compositions may fail to work well if there is only one single "bad" service used. A non-perfect but arguably useful instrument to inform about experienced service quality is to use consumer evaluation systems such as those introduced by online retailers. Resting on an intrinsic motivation of customers to rate products, such systems may give a valuable tool for deciding which service to request or to integrate into a composition.

Taking a theoretical perspective, we addressed two questions: First, how do providers react to rating systems in the sense that they may exploit a good reputation? Second, given information on ratings for composed services, can we use it to derive a rating of the component services?

In [MFHR18] we modify the model from [Del05] and model the situation of a service provider who strategically decides to repeatedly sell its service to customers in either high or low quality. The delivered service is rated as good or bad by the customer, who is, however, not fully able to identify the true quality. We therefore model a customer's feedback as a random variable, whose distribution depends on the delivered quality. Phrased differently, from the provider's perspective, there is a higher chance to receive a good rating when

the quality is good, compared to the case in which the quality is bad. Still, even with bad quality, the provider may receive a positive rating. The collection of the three most recent ratings is taken as a proxy for the provider's reputation. The service price is modeled to be directly dependent on the reputation, more precisely, on the number of positive ratings among the three most recent ratings. When making a strategic decision, the service provider has to compare the benefits from maintaining (or building) a good reputation with a higher sales price and higher revenues with the temptation of milking a good reputation by delivering bad quality and running the risk that prices will fall in response to a decline in reputation.

We address this trade-off by analyzing the theoretical model as well and testing the result in an experiment. Theoretically, we analyze the corresponding Markov Decision Problem and demonstrate that keeping the service quality constant is an optimal strategy for the service provider. In essence, whether supplying high or low quality is optimal depends on the difference of probabilities for receiving a positive rating. The larger the difference, the more accurate the consumer's rating is. This is intuitive because increasing the probability for a good rating when the quality is actually low increases the incentives to produce low quality.

Milking behavior means that the provider delivers bad quality whenever enough positive ratings appear in the rating history in order to keep the price high. When too many negative ratings appear, a good reputation is built up by delivering good quality which comes with a higher probability for good ratings. In the theoretical model the optimal strategy is either to constantly sell good or bad quality, so that milking one's own reputation does not take place. In the experiment, however, milking behavior can be observed, meaning that subjects with a high reputation tend to produce low quality for a couple of rounds, instead of maintaining the good reputation (which would have been the optimal solution). The striking lesson that we learn from this work impacts the design of reputation systems: The better the accuracy of the system, the higher the incentives to serve the market with good quality.

The second question focuses on how much we can say about the quality of single services. In [FHSS18] we discuss how to disentangle the ratings for service compositions from m services by n consumers as follows. Starting with the collection of user ratings over compositions, there has to be an aggregation step A and a disaggregation step D to arrive a rating over services, which leaves two options: a) either we first aggregate the ratings across users, which gives us an overall rating of compositions and we can then elicit (or disaggregate) information on single service ratings, or b) we first disaggregate individual ratings to individual ratings over single services, which should then be aggregated to an overall rating of single services. Figure 6 illustrates the aggregation/disaggregation problem. The starting point (see upper left corner) is a $n \times 2^m - 1$ matrix, in which each row corresponds to a user's evaluation of the $2^m - 1$ possible service compositions. The final result is supposed to be a $1 \times m$ matrix (lower right corner) that contains ratings for the m single services.

The task is to design informative aggregation operators A_1 and A_2 as well as disaggregation operators D_1 and D_2. While from the outset it is not evident which operators fit best, one essential property we impose is that the two routes sketched above yield the same result, making the diagram in Figure 6 commutative. Moreover, anonymity requirements guarantee that no single service and no two users are treated differently. Further, it should

$$
\begin{array}{ccc}
\mathcal{M}\,(n \times 2^m - 1) & \xrightarrow{\;\;D_1\;\;} & \mathcal{M}\,(n \times m) \\[2pt]
\downarrow{\scriptstyle A_1} & \xrightarrow[\;\;D_2\;\;]{} & \downarrow{\scriptstyle A_2} \\[2pt]
\mathcal{M}\,(1 \times 2^m - 1) & & \mathcal{M}\,(1 \times m)
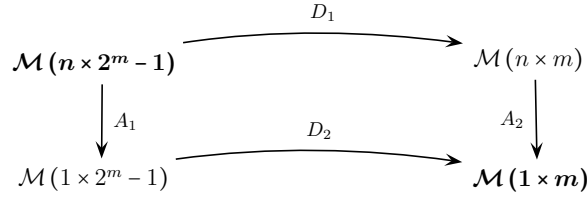\end{array}
$$

Figure 6: *Aggregation and disaggregation (from [FHSS18]).*

not be possible for a single user to significantly manipulate the rating of a single service by changing its own valuations.

For the disaggregation step, we reinterpret a rating over compositions as a cooperative game with transferable utility (TU game) and use the Shapley value as a solution concept to attribute a rating to each single service/player. For an aggregation device (across users), we use the averaging operator. This combination turns out to be commutative in the sense above. It is anonymous and no single user has a strong influence on the final valuations. Other (intuitive) methods such as taking minimal or maximal composition values particularly fail to satisfy this non-manipulability property.

For the OTF market, this means that a smartly designed system that processes end user valuations over composed services can help to identify those (component) services that fail to work well in compositions. This information in turn can be used by OTF service providers when composing services to satisfy a user request. Therefore, the demand for and pricing of services are influenced by the analysis of customer feedback.

Mechanism Design

Designing a market means designing the rules for interaction according to which the market participants react to and choose their strategies. The combination of strategies determine allocations, payoffs, or welfare, hence the market outcome. The design of rules such that strategic behavior finally leads to a desired (market) outcome is at the heart of implementation theory or mechanism design. At a fundamental level, the question arises which outcomes can be implemented through strategic interaction at all and what is an appropriate equilibrium concept. In [HT21] we analyze this problem in a very general model. The notion of a mechanism (describing the rules of interaction) is expanded to one of a *socio-legal system*, which allows to cope with two types of obstacles that have been widely ignored in the mechanism design literature.

First, unlike in traditional mechanism design, a player's set of feasible strategies may depend on the other players' choices of strategy. As a consequence, specific strategy profiles might not be feasible. For a simple illustrative example, consider a number of OTF service providers who choose how much capacity of a hardware resource they want to use. Because the total capacity is limited, each strategy profile of the other providers sets an upper bound for the choice of a particular provider. Phrased differently, it might occur that the total capacity chosen by all providers exceeds the maximally possible capacity of the resource. However, one of the providers could be blamed for choosing the "wrong" strategy, because feasibility is a property of the chosen profile of strategies. Second, the mechanism designer (e.g., the OTF market maker) might want to avoid "illegal" behavior in the sense that particular outcomes should not occur. As a simple example one can think

of OTF service providers who can choose to either serve or not serve a particular customer. The designer can declare that all profiles in which at least one provider serves the customer are "legal" ones, because he is interested in an outcome guaranteeing that the customer is served.

The extension from mechanisms to socio-legal systems requires an adaption of the equilibrium concept. We define the notion of a *Debreu-Hurwicz equilibrium* that combines the Nash equilibrium concept with features from Hrwicz' work on legality and Debreu's social equilibrium concept (see [Hur94], [Deb52], [KY18]). The choice or design of the equilibrium concept is ultimately linked to the question which social choice rules, i.e., which desirable outcomes, are implementable.

We address this question by investigating implementability of the cooperative Nash bargaining solution, which marks the desired outcome for a population of players. We find that by gradually expanding the equilibrium notion from Nash to Debreu-Hurwicz, undesired equilibria (i.e., those that do not trigger the Nash bargaining solution as an outcome) can be removed so that we have ultimately been able to show a new uniqueness result. Although our paper does not directly construct or analyze a specific interaction in the OTF market, it aims at opening a new route in implementation theory allowing to explicitly cope with unwanted behavior by the players. This is important for the functionality of interaction.

2.3   Algorithmic Game Theory

We now review the primary works in subproject A3 that employ the algorithmic game theory approach. Game theory complements the mechanism design approach, where one designs interaction, by rather analyzing, e.g., existence and properties of equilibria in specific interactions. Algorithmic game theory studies both interactions of algorithms, such as the competition of trading or negotiation algorithms, and also algorithms executed on models of interactions, such as computing Nash equilibria or bargaining outcomes.

We start off by presenting general models of strategic sharing of resources under the umbrella of budget games, which model a market of products, and of various congestion games, which model sharing situations. We then continue with the less general but more network-specific progressive filling games, where the choice of routes determines the allocated bandwidth in a natural utility max-min fairness manner. Finally, we present a practically relevant online algorithm generalizing bin packing.

Sharing of resources
Our works contribute to the literature on sharing resources. [DRS14] study a market situation via introducing *budget games*, in which players choose tasks (products), that in turn have demands for resources. Consequently, choices have an influence on the sharing of necessary resources between chosen tasks. The budgets of resources are either shared proportionally between the tasks or dependent on the decision order. The authors studied the optimal solution, as well as the existence, complexity and efficiency of equilibria.

This model, for instance, describes resource sharing that occurs in cloud computing where the clients compete on the products of the cloud. In the strategic variant of the game, in which market entrance is simultaneous, the utility of a resource is shared proportionally. In

contrast, in an ordered budget game that models the market entrance order the resources are allocated in the entrance order, and a deviator moves to the last position. Since the strategic budget game is a basic utility game, its price of anarchy is at most 2, as proven in [Vet02]. [DRS14] prove that this bound also holds for ordered budget games. First, [DRS14] prove that finding the optimal allocation is NP-hard and can be approximated within $1\breve{\ }1/e$, provided the players' strategies form a matroid. Concerning the Nash equilibrium, it may not exist in the strategic budget game and deciding whether it does or not is NP-hard. In the ordered budget games, even strong Nash equilibria exist and are polynomially computable. The strong price of stability is 1, while the strong price of anarchy is 2. The authors demonstrate that improvement moves converge to a Nash equilibrium, but it may take exponentially many steps.

A related model, which also studies sharing resources, albeit differently, is the one of *congestion games and their variations*, which capture many important interactions: in particular, network interactions where bandwidth, CPU or another resource is used by several parties. In the face of the need imposed by their ubiquity, computing the equilibria of congestion games is appallingly PLS-complete ([FPT04; AS08; ARV08]). Moreover, weighted congestion games may possess no potential function or even no pure Nash equilibria at all ([GMV05; FKS05]), and it is NP-hard to decide whether Nash equilibria exist ([DS08]). The only classes where a potential always exists are classes with linear or exponential cost functions ([FKS05; HK12; PS07]). But even for linear costs, computing and equilibrium is PLS-complete ([ARV08]). Since mixed equilibria are generally harder to interpret, the lack of pure ones indeed poses a problem. In order to ameliorate the existence and computation problem, [CFGS15] studies existence and structure of approximate Nash equilibria in weighted congestion games. They also proposed several algorithms to find approximate Nash equilibria. An earlier algorithm by the same authors in [CFGS11] computes a constant-approximate Nash equilibrium in *unweighted* congestion games with cost functions all being constant-degree polynomials. Another known result is that for symmetric unweighted congestion games, any $1 + \epsilon$-improvement dynamics converges to a $1 + \epsilon$-approximate Nash equilibrium in a polynomial number of steps ([CS11]). Moreover, [AAE+08] shows rapid convergence to socially efficient states, but those states need not be approximate equilibria.

[CFGS15] approximates a given weighted potential game with a special potential game termed Ψ-*game*. They approximate a weighted congestion game with cost functions of degrees at most $d \geq 2$ with a Ψ-game of degree d, and prove that a ρ-approximate equilibrium of such a Ψ-game of degree d constitutes a $d!\rho$-approximate equilibrium of the original weighted congestion game. Since the Ψ-games have potential and thus a Nash equilibrium, this implies that the original game possesses a $d!$-approximate equilibrium. They also provide polynomial approximation algorithm for constant d and bound the length of a best-response sequence from any initial state to a $d^{O(d^2)}$-approximate pure Nash equilibrium.

Following up on [CFGS15], [HKS14] sets out to improve the approximation factors of approximate pure Nash equilibria. Since the existence of an α-approximate potential function implies the existence of an α-approximate Nash equilibrium and the convergence to such an equilibrium of steps improving by the factor of at least α, they concentrate on α-approximate potential functions with smallest possible αs. For several cost functions, such as the polynomial ones or the concave ones, they prove the existence of α-approximate

potential functions with smaller values of α than was previously known. Concretely, they provide the upper bounds of 3/2 for concave cost functions and bounds of 4/3, 1.785 and 2.326 for polynomials of degrees 2, 3 and 4, respectively. In general, for polynomials of degree l, their bound is $l + 1$. For two players, their results are provably tight.

Progressive filling games

Congestion games constitute very important general models, but they assume a player's bandwidth is the sum of what she gets allocated on each edge, rather than the maximum thereof. This is ameliorated by the bottleneck congestion games ([CDR06]) where the bandwidth of a player is the maximum allocated bandwidth. However, the computation of an equilibrium there is NP-hard. Moreover, the main modeling disadvantage of bottleneck congestion games is the lack of flexibility in bandwidth allocation, contrary to the flexible *Max-Min Fairness (MMF)* from [BG21], which we present next. Therefore, [HHSS14] defines and analyzes *Progressive Filling Games (PFG)*, which model players choosing routes and receiving fair bandwidth according to the MMF algorithm. That work generalizes [YXF+10; YXF+13] to strong NE and a broader class of water-filling algorithms. They also provide a picture of the complexity of computing SNE and present the prices of anarchy and stability, as we now describe. MMF is a known fairness standard, where nobody's allocation can be increased without hurting a worse-off party. Some known generalizations include weighted MMF and utility MMF. This paper implements utility MMF by a polynomial water-filling algorithm. They define routing games with progressing filling, where each player picks a set of resources, aiming to optimize her allocated bandwidth. They assume that the flow control instantly converges to the corresponding generalization of MMF after each route update, an assumption justified, for example, by [WLLD05].

[WLLD05] studies the existence, the computation and the efficiency of the pure and of strong NE in these games. They first prove the existence of strong NE for any generalization of the water-filling algorithm. As long as certain conditions on the rate functions hold, conditions that cannot be dropped. The authors also suggest an algorithm to compute a strong NE, employing a packing oracle. They then present hardness results for computing strong NE. Next, the authors provide tight bounds on the prices of anarchy and stability, assuming the utilitarian social welfare, providing bounds that hold even if an arbitrary capacity-respecting allocation is allowed, not necessarily an MMF one. In general, the prices of anarchy and stability are n, and this is tight for both pure and strong equilibria. For routing a single commodity using MMF, the price of stability is $2 - (1/n)$ for both normal and strong NE, and the price of anarchy is n for NE and 4 for strong NE, all the bounds besides the latter being tight. If the allocation rules are fixed, the $2 - (1/n)$ bound cannot be overcome. However, if we can adjust the weights in the weighted MMF water-filling algorithm, then we can make the game have an optimum SNE. This is NP-hard to compute, but can be approximated.

Bin packing

Since execution of composed services is an integral part of the OTF market, an important algorithmic topic addressed in subproject A3 is cloud-server storage, balancing the server limitations and minimizing the cloud costs, including reducing the wear and tear of the server, energy costs, and the communication and the execution time. This is modeled in [FFG+18], who design an *online algorithm for dynamic bin packing* that balances

competitiveness ratio with minimizing the number of repacks, called recourse. Offline bin packing is approximable with additive $O(\log OPT)$, where OPT is the optimum value, but online bin packing has a 1.540 multiplicative gap, even when the optimum value approaches infinity. This work now considers fully dynamic bin packing with bounded recourse, namely allowing arrival and departures of items and their repacking. They define worst-case and amortized recourse, measuring the movement costs at each time or in total, respectively.

This work characterizes the recourse to asymptotic competitive ratio trade-off in the following cases. For unit movement costs, they provide tight upper and lower bounds. The asymptotic competitive ratio here is better than that for online bin packing without repacking! That technique uses LPs and Myopic packing from [Ivk95]. For general movement costs, [Sei02] suggests a super harmonic algorithm with constant recourse, implying a competitive ratio of 1.589, which is close to the best known bin packing result of 1.578 [BBD+18]. Moreover, the authors conjecture that fully dynamic algorithms can be reduced to online ones, which would imply equal asymptotic competitive ratios, while maintaining a constant recourse. Finally, if the costs are just equal to the sizes, the authors provide a tight bound.

3 Conclusion and Outlook

The lessons we learned from our analyses of the OTF market and the interactions within raise new questions that are not exclusively interesting for the functioning of OTF markets. In what follows, we briefly comment on these more general implications of our findings for economic modeling and future works.

Bundling

The theory of bundling and tying is highly developed. Our paper is unique in that it specifically takes and analyzes an asymmetric exogenous market and distribution structure as a starting point. What is more, we show that under certain conditions this asymmetric distribution structure emerges as an equilibrium outcome in a richer model where the participating service providers also decide on their distribution channels.

Our question, under which conditions bundling occurs, does not only represent one of the core questions in the analysis of OTF markets, but also has important bearings for the analysis of bundling and tying in general markets. In particular, our findings point to negative welfare effects of product bundling and raise serious antitrust concerns. An empirical analysis to quantify the welfare effects would be desirable.

Opening markets for competition

Politicians and economists alike quite generally believe in the benefits of competition. Accordingly, they often advocate the opening of markets for competitors. This issue has been thoroughly analyzed for 'normal' markets, but not so much for public provider monopolies that are opened for competition to private providers. Here, the contribution of our paper sets in.

Our experimental analysis points both to beneficial effects that are caused by introducing competition, but also to negative effects that result from collusion and from the immobility

(or non-responsiveness) of customers. Importantly, introducing competition amplifies treatment inequalities across customer groups.

Our theoretical analysis shows that introducing competition in a price-regulated market does not necessarily bring about higher quality. In particular, if the regulator's budget is small, opening a formerly monopolistic market for competition will entail that an entrant differentiates its product away from the existing product in order to soften competition. As a consequence quality may not be raised at all. For larger budgets, quality will eventually be raised by both the incumbent public provider and the private entrant. However, these budgets also need to be financed, the cost of which is substantial.

Our findings raise a number of interesting questions. As to our experimental analysis, it would be instrumental to identify ways that would allow to avoid the unequal treatment of customers, in particular the low quality treatment of disadvantaged ones. Our theoretical model could be extended to allow for multiple dimensions of quality (contractible vs. non-contractible) or to include horizontal dimensions other than location.

Firm survival and innovation
Our economic evolutionary approaches to the analysis of behavior in innovation contest represent the first theoretical approaches that are capable of explaining overdissipation, a phenomenon that has also been observed empirically in experiments. It would be interesting to engage in an empirical analysis that covers real markets for innovation. Here, a structural econometric approach would be appropriate.

Bargaining
The interaction among few players requires an alternative to models of competition. Instead, bargaining models are the more appropriate choice. While fairness and efficiency are compatible in the case in which players are completely informed about each other's preferences, we have identified a trade-off between these properties in the case of incomplete information. Still, in the literature on bargaining theory there are few works that actually work out this trade-off by rigorously defining and analyzing solution concepts that are adapted to the information scenario. Especially asymmetric solutions deserve a more detailed investigation, because not only in the OTF context do the players bring different skills or power to the bargaining table.

To better understand differences between the various solution concepts that are discussed in the literature, it would be worthwhile to find a common ground in the sense of a unified approach. This could be done either on a descriptive basis, so that solutions appear as special cases of a more "universal" solution, or on a normative ground, meaning that characterizing axioms are comparable. Either way, results may help to judge which bargaining solution is most appropriate for a given problem.

Matching
Matching theory is a vital and growing research field that has branched out into many diverse applications and questions. From our works on matching mechanisms we have learned that large markets, in which agents are naturally not able to know all other agents and their preferences, require a careful design of matching mechanisms that provide the right incentives.

When encountering the largeness of a market by (partly) generating preferences, particular patterns may occur that cause unwanted incentives for strategic behavior. Thus, we see a tradeoff between facilitation and prevailing incentives, which has not yet been well explored in the literature.

The treatment of primary data on preferences marks another challenge for future research. Since matching mechanisms are typically conducted by a central clearinghouse, all relevant information has to be collected at one place. A distributed version of a mechanism that collects information at different places contributes to the protection of private data and thus enhances the acceptability of the procedure. Here, cryptographic methods may help to reach decentralized versions with a minimum exchange of information.

Reputation systems

We have seen that implementing a rating system for services that is as accurate as possible reduces the incentives for providers to milk one's own reputation and deliver poor quality services at a high price. Further, a smart processing of user evaluations can identify poor quality. Both results are promising, not only in the light of OTF markets. The study of aggregation and disaggregation of user information can still be refined in a positive as well as a normative direction. From a positive viewpoint, for example, the investigation would look into how an inherent asymmetry among users (experts vs. ordinary end users) can be reflected in the aggregation process. From a normative viewpoint, a characterization of data processing through axioms is still missing with which ideally different mechanisms can be assessed by their defining properties.

Apart from theoretical work, it will be interesting to investigate how real agents react to different designs of a reputation system. As a result, experimental evidence gives (further) advice which elements of a system are important. This includes anonymity of raters, the rating scale, or (possibly aggregated) information that is displayed.

Mechanism design

Here, our research is meant to be of a more general nature. The aim was to describe a framework in which unwanted strategic behavior can be captured. In the language of OTF markets, one could say that the market provider (the mechanism designer) may use mechanisms that, e.g., rule out the use of an illegal strategy. At a conceptual level, this means that the responsibility for adhering to the rules is shifted from the players to the designer. However, this immediately renews an old question about the responsibility of the designer or, as cited in [Hur94], "but who will guard the guardians." Fitting the idea that OTF markets are self-organizing systems, one way could be to equip a mechanism with a control device that allows the players themselves to regulate the designer's choice of mechanism.

Bibliography

[AAE+08]   AWERBUCH, B.; AZAR, Y.; EPSTEIN, A.; MIRROKNI, V. S.; SKOPALIK, A.: Fast Convergence to Nearly Optimal Solutions in Potential Games. In: *Proceedings of the 9th ACM Conference on Electronic Commerce*. EC '08. Chicago, Il, USA: Association for Computing Machinery, 2008, pp. 264–273.

[ARV08]   ACKERMANN, H.; RÖGLIN, H.; VÖCKING, B.: On the Impact of Combinatorial Structure on Congestion Games. In: *J. ACM* 55 (Dec. 2008), no. 6.

[AS03]   ABDULKADIROĞLU, A.; SÖNMEZ, T.: School choice: A mechanism design approach. In: *The American Economic Review* 93 (2003), no. 3, pp. 729–747

[AS08]   ACKERMANN, H.; SKOPALIK, A.: Complexity of Pure Nash Equilibria in Player-Specific Network Congestion Games. In: *Internet Mathematics* 5 (2008), no. 4, pp. 323–342

[BBD+18]   BALOGH, J.; BÉKÉSI, J.; DÓSA, G.; EPSTEIN, L.; LEVIN, A.: A New and Improved Algorithm for Online Bin Packing. In: *26th Annual European Symposium on Algorithms (ESA 2018)*. Ed. by AZAR, Y.; BAST, H.; HERMAN, G. Vol. 112. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:14.

[BG21]   BERTSEKAS, D.; GALLAGER, R.: *Data Networks: Second Edition*. Athena Scientific, 2021.

[BHK17]   BROSIG-KOCH, J.; HEHENKAMP, B.; KOKOT, J.: The effects of competition on medical service provision. In: *Health Economics* 26 (2017), no. 53, pp. 6–20

[BHK23]   BROSIG-KOCH, J.; HEHENKAMP, B.; KOKOT, J.: Who benefits from quality competition in health care? A theory and a laboratory experiment on the relevance of patient characteristics. In: *Health Economics* (2023)

[BM14]   BIRÓ, P.; MCDERMID, E.: Matching with sizes (or scheduling with processing set restrictions). In: *Discrete Applied Mathematics* 164 (2014), pp. 61–67

[CDR06]   COLE, R.; DODIS, Y.; ROUGH2006GARDEN, T.: Bottleneck Links, Variable Demand, and the Tragedy of the Commons. In: vol. 60. Jan. 2006, pp. 668–677

[CFGS11]   CARAGIANNIS, I.; FANELLI, A.; GRAVIN, N.; SKOPALIK, A.: Efficient Computation of Approximate Pure Nash Equilibria in Congestion Games. In: *CoRR* abs/1104.2690 (Apr. 2011)

[CFGS15]   CARAGIANNIS, I.; FANELLI, A.; GRAVIN, N.; SKOPALIK, A.: Approximate Pure Nash Equilibria in Weighted Congestion Games: Existence, Efficient Computation, and Structure. In: *Transactions on Economics and Computation* 3 (2015), no. 1

[CLS16]   CHEN, Y.; LIANG, Y.; SÖNMEZ, T.: School choice under complete information: An experimental study. In: *The Journal of Mechanism and Institution Design* 1 (2016), no. 1, pp. 45–82

[CS06]   CHEN, Y.; SÖNMEZ, T.: School Choice: an experimental study. In: *Journal of Economic theory* 127 (2006), no. 1, pp. 202–231

[CS11]   CHIEN, S.; SINCLAIR, A.: Convergence to approximate Nash equilibria in congestion games. In: *Games and Economic Behavior* 71 (2011), no. 2, pp. 315–327.

[Deb52]   DEBREU, G.: A social equilibrium existence theorem. In: *Proceedings of the National Academy of Sciences* 38 (1952), no. 10, pp. 886–893

[Del05]   DELLAROCAS, C.: Reputation Mechanism Design in Online Trading Environments with Pure Moral Hazard. In: *Information Systems Research* 16 (2005), no. 2, pp. 209–230

[DRS14]   DREES, M.; RIECHERS, S.; SKOPALIK, A.: Budget-Restricted Utility Games with Ordered Strategic Decisions. In: *Algorithmic Game Theory - 7th International Symposium, SAGT 2014, Haifa, Israel, September 30 - October 2, 2014. Proceedings*. Ed. by LAVI, R. Vol. 8768. Lecture Notes in Computer Science. Springer, 2014, pp. 110–121.

[DS08]   DUNKEL, J.; SCHULZ, A. S.: On the Complexity of Pure-Strategy Nash Equilibria in Congestion and Local-Effect Games. In: *Mathematics of Operations Research* 33 (2008), no. 4, pp. 851–868.

[EHH22]   ENDRES, A. E.; HEHENKAMP, B.; HEINZL, J.: *The Impact of Product Differentiation on Retail Bundling in a Vertical Market*. Tech. rep. Paderborn University, 2022

[FFG+18]   FELDKORD, B.; FELDOTTO, M.; GUPTA, A.; GURUGANESH, G.; KUMAR, A.; RIECHERS, S.; WAJC, D.: Fully-Dynamic Bin Packing with Little Repacking. In: *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Ed. by CHATZIGIANNAKIS, I.; KAKLAMANIS, C.; MARX, D.; SANNELLA, D. Vol. 107. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 51:1–51:24.

[FHSS18]   FELDOTTO, M.; HAAKE, C.-J.; SKOPALIK, A.; STROH-MARAUN, N.: Disaggregating User Evaluations Using the Shapley Value. In: *Proceedings of the 13th Workshop on Economics of Networks, Systems and Computation (NetEcon 2018)*. Irvine, California, USA, 2018, 5:1–5:6

[FKS05]   FOTAKIS, D.; KONTOGIANNIS, S.; SPIRAKIS, P.: Selfish unsplittable flows. In: *Theoretical Computer Science* 348 (2005), no. 2. Automata, Languages and Programming: Algorithms and Complexity (ICALP-A 2004), pp. 226–239.

[FPT04]   FABRIKANT, A.; PAPADIMITRIOU, C.; TALWAR, K.: The Complexity of Pure Nash Equilibria. In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*. STOC '04. Chicago, IL, USA: Association for Computing Machinery, 2004, pp. 604–612.

[GHL19]   GU, Y.; HEHENKAMP, B.; LEININGER, W.: Evolutionary equilibrium in contests with stochastic participation: Entry, effort and overdissipation. In: *Journal of Economic Behavior and Organization* (2019), pp. 469–485

[GMV05]   GOEMANS, M.; MIRROKNI, V.; VETTA, A.: Sink equilibria and convergence. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 2005, pp. 142–151

[GS62]   GALE, D.; SHAPLEY, L. S.: College admissions and the stability of marriage. In: *The American Mathematical Monthly* 69 (1962), no. 1, pp. 9–15

[HHSS14]   HARKS, T.; HÖFER, M.; SCHEWIOR, K.; SKOPALIK, A.: Routing Games with Progressive Filling. In: *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*. 2014, pp. 352–360

[HK12]   HARKS, T.; KLIMM, M.: On the Existence of Pure Nash Equilibria in Weighted Congestion Games. In: *Mathematics of Operations Research* 37 (2012), no. 3, pp. 419–436.

[HK20]   HEHENKAMP, B.; KAARBØE, O. M.: Location Choice and Quality Competition in Mixed Hospital Markets. In: *Journal of Economic Behavior and Organization* 177 (2020), pp. 641–660

[HK23]   HEHENKAMP, B.; KAARBØE, O. M.: *Price Regulation, Quality Competition and Location Choice with Costly Relocation*. Tech. rep. Paderborn University, 2023

[HKS14]   HANSKNECHT, C.; KLIMM, M.; SKOPALIK, A.: Approximate pure Nash equilibria in weighted congestion games. In: *Proceedings of the 17th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. 2014, pp. 242–257

[HLP04]   HEHENKAMP, B.; LEININGER, W.; POSSAJENNIKOV, A.: Evolutionary equilibrium in Tullock contests: spite and overdissipation. In: *European Journal of Political Economy* 20 (2004), no. 4, pp. 1045–1057.

[Hoo20]   HOOF, S.: On a class of linear-state differential games with subgame individually rational and time consistent bargaining solutions. In: *Journal of Mechanism and Institution Design* 5 (2020), p. 1

[HR18]   HAAKE, C.-J.; RECKER, S.: The Generalized Nash Bargaining Solution for Transfer Price Negotiations under Incomplete Information. In: *Group Decision and Negotiation* 27 (2018), no. 6, pp. 905–932

[HS18]   HAAKE, C.-J.; STROH-MARAUN, N.: Outcome equivalence in school choice with reciprocal preferences. In: *Economics Letters* 170 (2018), pp. 39–41

[HS20a]   HOYER, B.; STROH-MARAUN, N.: *Matching strategies of heterogeneous agents under incomplete information in a university clearinghouse*. Tech. rep. 2020, pp. 453–481

[HS20b]   HOYER, B.; STROH-MARAUN, N.: *Stability in Weighted College Admissions Problems*. Working Papers Dissertations 63. Paderborn University, Faculty of Business Administration and Economics, May 2020.

[HT21]   HAAKE, C.-J.; TROCKEL, W.: Socio-legal systems and implementation of the Nash solution in Debreu–Hurwicz equilibrium. In: *Review of Economic Design* (2021)

[Hur94]   HURWICZ, L.: Economic design, adjustment processes, mechanisms, and institutions. In: *Economic Design* 1 (1994), no. 1, pp. 1–14

[Ivk95]   IVKOVIĆ, Z.: *Fully dynamic approximation algorithms*. University of Delaware, 1995

[KY18]   KORAY, S.; YILDIZ, K.: Implementation via rights structures. In: *Journal of Economic Theory* 176 (2018), pp. 479–502

[MFHR18]   MIR DJAWADI, B.; FAHR, R.; HAAKE, C.-J.; RECKER, S.: Maintaining vs. Milking Good Reputation when Customer Feedback is Inaccurate. In: *PLoS ONE* 13 (2018), no. 11

[MM10]   MCDERMID, E. J.; MANLOVE, D. F.: Keeping partners together: algorithmic results for the hospitals/residents problem with couples. In: *Journal of Combinatorial Optimization* 19 (2010), no. 3, pp. 279–303

[PS07]   PANAGOPOULOU, P. N.; SPIRAKIS, P. G.: Algorithms for Pure Nash Equilibria in Weighted Congestion Games. In: *ACM J. Exp. Algorithmics* 11 (Feb. 2007), 2.7–es.

[PS08]   PATHAK, P. A.; SÖNMEZ, T.: Leveling the playing field: Sincere and sophisticated players in the Boston mechanism. In: *The American Economic Review* 98 (2008), no. 4, pp. 1636–1652

[Rot89]   ROTH, A. E.: Two-sided matching with incomplete information about others' preferences. In: *Games and Economic Behavior* 1 (1989), no. 2, pp. 191–209

[Sei02]   SEIDEN, S. S.: On the Online Bin Packing Problem. In: *J. ACM* 49 (Sept. 2002), no. 5, pp. 640–671.

[SS74]   SHAPLEY, L.; SCARF, H.: On cores and indivisibility. In: *Journal of Mathematical Economics* 1 (1974), no. 1, pp. 23–37

[Str20]   STROH-MARAUN, N.: *Pareto Efficiency in Weighted School Choice Problems*. Working Papers Dissertations 64. Paderborn University, Faculty of Business Administration and Economics, May 2020.

[Vet02]   VETTA, A.: Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* 2002, pp. 416–425

[Wag94]   WAGENHOFER, A.: Transfer pricing under asymmetric information: An evaluation of alternative methods. In: *European Accounting Review* 3 (1994), no. 1, pp. 71–103

[WLLD05]   WANG, J.; LI, L.; LOW, S.; DOYLE, J.: Cross-layer optimization in TCP/IP networks. In: *IEEE/ACM Transactions on Networking* 13 (2005), no. 3, pp. 582–595

[YXF+10]   YANG, D.; XUE, G.; FANG, X.; MISRA, S.; ZHANG, J.: Routing in max-min fair networks: A game theoretic approach. In: Oct. 2010, pp. 1–10

[YXF+13]   YANG, D.; XUE, G.; FANG, X.; MISRA, S.; ZHANG, J.: A Game-Theoretic Approach to Stable Routing in Max-Min Fair Networks. In: *IEEE/ACM Transactions on Networking* 21 (2013), no. 6, pp. 1947–1959

Subproject A4:
Empirical Analysis in Markets for OTF Services

Behnud Mir Djawadi[1], Alina Elrich[1], René Fahr[1], Bernd Frick[1], Daniel
Kaimann[1], Dennis Kundisch[1], Michelle Müller[1], Martin Poniatowski[1],
Sabrina Schäfers[1]

1 Faculty of Business Administration and Economics,
Paderborn University, Paderborn, Germany

1 Introduction

In OTF markets, sophisticated combined services in mainly small quantities with predominantly experience attributes are traded. These attributes, which can hardly be known before using the service, and the many actors involved in creating these services result in unavoidable information asymmetries with corresponding adverse effects on the market outcome, including the risk of market failure [ake78]. Due to these characteristics of the OTF market, online reviews and certificates are significant in reducing information asymmetries between the service provider and the customer about the quality of the traded services. While a substantial body of literature emerged that examines to what extent online reviews are an effective measure to mitigate this problem, several research gaps are identified for the study of online reviews in OTF markets.

A) Influence of service and market characteristics on online reviews and economic outcomes

Understanding the influence of service and market characteristics on online reviews and economic outcomes is crucial for developing a successful OTF market in which all participants can generate (economic) benefits. Prior research has revealed that an increase in the average rating, as well as the total number of reviews from consumers, have a positive effect on the demand of a product or service and the price offered by a firm (e.g.,[LH08]). Moreover, several studies have estimated the to what extent reviews by professional critics impact the sales performance [HMM12]. However, a significant limitation of many of these studies is that they tend to control either only for consumer reviews or only for professional critics, even though Chintagunta et al. [CGV10] have highlighted the potentially significant differences between reviews from professional critics and users. Therefore, as a part of the CRC, Cox and Kaimann [CK15] analyze the relationship between economic outcomes and two signals of product quality: reviews from professional critics and reviews from

behnud.djawadi@wiwi.uni-paderborn.de (Behnud Mir Djawadi), alina.elrich@uni-paderborn.de (Alina Elrich), rene.fahr@wiwi.uni-paderborn.de (René Fahr), bernd.frick@wiwi.upb.de (Bernd Frick), daniel.kaimann@uni-paderborn.de (Daniel Kaimann), dennis.kundisch@wiwi.uni-paderborn.de (Dennis Kundisch), michelle.mueller@wiwi.uni-paderborn.de (Michelle Müller), martin.poniatowski@wiwi.uni-paderborn.de (Martin Poniatowski), sabrina.schaefers@uni-paderborn.de (Sabrina Schäfers)

consumers. Additionally, as the current literature has only partially examined the influence of ratings and ad expenditures separately and jointly on a product's or service's demand in online markets, Frick and Kaimann [FK17] shed light on this relationship. Moreover, what is less well understood in the online review literature are the effects of the variance of the online reviews (i.e., the distribution of the online ratings) on economic outcomes on the level of firms. To address this gap, Zimmermann et al. [ZHKN18] developed an analytical model that decomposes the variance of online ratings into two sources: *taste differences* in search and experience attributes of a product or service and *quality differences* among instances of this product or service in the form of product failure. In addition, on the more aggregated market level, it has remained unclear what impact the local market competition has on the heterogeneity of available businesses in a market. Gutt et al. [GHR19] contributed to the answer to this question by empirically investigating the relationship between the range and average of the mean rating distribution in the market and market competition.

B) Opportunistic behavior of service providers

Complex services are combined in OTF markets. Customers may not have the expertise to judge the service or product quality correctly. In OTF markets, there is the potential that customers may make mistakes in evaluating the true quality of the provided service. This inaccurate customer feedback, however, may impact the service provider's decisions about the quality. For example, the service provider might be tempted to charge a high price but deliver low quality, as the customer erroneously may rate the service as high quality. While the current literature has addressed the effect of inaccurate customer feedback on the functioning of reputation systems (e.g., [MDC14]), it only focused on intentional customer behavior. For example, customers intentionally fake transactions and provide dishonest feedback to damage the seller's or service provider's reputation. The study by Mir Djawadi et al. [MFHR18] closes this research gap by examining how inaccurate customer feedback based on unintentional mistakes influences the strategic quality choices of a service provider. As a result of this, the focus lies on the behavior of the service providers, whether they seek to maintain a good reputation by providing high-quality services, even if it is tempting that the customer might not detect low quality, or whether they milk a good reputation and exploit the customers.

C) Design of online review systems in OTF markets

For some design elements, literature already shows how design decisions influence the effect of drivers of writing a review (e.g. self-expression, altruism, or a sense of belonging) on online ratings. For example, the number of evaluation dimensions is related to the evaluation level. However, there are significant research gaps in the impact and potential of design decisions in OTF-like markets. Such design decisions include metrics, the degree of anonymity, or social proximity between trading partners. Metrics (i.e., aggregated measurements of customer feedback) enable the evaluation of product quality and the comparability of products on marketplaces. Since cognitive biases identified by psychology and behavioral economics affect people's perception and way of thinking, it is unclear whether the technically implemented aggregation functions correspond to customers' actual aggregation behavior. To address this, van Straaten et al. [SMH+21] elicit customers' aggregation heuristics and contrast these with reference functions.

Anonymous reputation systems can be implemented to avoid concerns about sharing private information by providing a customer review. However, this anonymity can potentially crowd out reviews motivated by customers' self-expression (one of the main motives for writing reviews). It is unclear how anonymity and its driving forces (such as blunting self-presentation) affect customers' propensity to write reviews [RS12]. Therefore, Hoyer and van Straaten [HS22] investigate whether anonymous reputation systems have the disadvantage of blunting self-presentation and whether altruistic attitudes moderate this effect.

A growing body of research looks at mutual evaluations and demonstrates that such situations typically result in valuations that are biased upward in magnitude. This, in turn, reduces market efficiency and, in the worst case, leads to market failures. Mir Djawadi and Wester (2022) [MW22] examine to what extent social proximity between trading partners leads to upward-biased ratings.

D) Certification to reveal service quality in OTF markets

Another essential instrument for reducing information asymmetries in markets for complex (combined) physical and digital services is a certification (such as certified consulting services or information security certification in online shopping). Certification by an external agent allows providers to signal (minimum) standards for the certified product or service characteristics, thus reducing information asymmetries.

While the effects of certifications are clear from a theoretical point of view, their empirical relevance has not been convincingly documented. Accordingly, the study by Fanasch and Frick [FF20] compares the effects of self-declaration and certification on product prices by distinguishing between certified products and service characteristics. The results of this study shed light on the unique relationship of certifications to reduce information asymmetries in OTF markets.

The composition of individual services causes additional complexity when analyzing service quality certification. In particular, it is unclear whether certification of all individual services or combined services is necessary. Consequently, Fanasch and Frick [FF20] provide evidence on whether and to what extent collective reputation for combined services impacts market prices.

2   Main Contributions

In the following, the main contributions are described in greater detail, which are meant to address the research gaps regarding the mitigation of information asymmetries in OTF markets introduced before.

2.1   Influence of Service and Market Characteristics on Online Reviews and Economic Outcomes

Interaction and impact of reviews from professional critics and customer reviews on the market performance of experience goods

Due to the digitalization of retail, customers can express their product experiences via user-generated reviews in various discussion forums, assessment portals, or on retailer websites. We contend that independent information expressed in product reviews has three plausible ways in which it can influence consumer behavior. First, more positive evaluations lower uncertainty over product quality among prospective consumers. Secondly, if evaluations reach near unanimity in opinion, then customers' certainty in their purchase decision will increase. Thirdly, the more reviews submitted, the lower the evaluation insecurity among future buyers. We focus on the relationship between the different facets of customer reviews: namely, valence, volume, and variance.

Valence represents the average weighted review score from customers. The use of weighted online average scores is consistent with prior studies of experience goods, e.g., movies, books, and magazines [CGV10]. Volume measures the total number of reviews posted. We expect that more significant numbers of ratings reduce the information asymmetry and thus positively affect consumer choices. Our measure of variance is the sum of squares of the proportion of positive, negative, and mixed opinions among the total number of reviews. It is thus analogous to the Hirschman-Herfindahl index, which captures the degree of homogeneity or heterogeneity of any variable of interest. Our variance variable is therefore bounded between an upper value of 1 (perfect homogeneity) and a lower value of 0.33 (perfect heterogeneity; equal proportions of reviews in each of the three opinion categories).

Cox and Kaimann [CK15] have focused on the relationship between commercial performance and two signals of product quality: reviews from professional critics and online word-of-mouth. By analyzing the consumer behavior and sales performance in the digital platform industry of video games, we shed light on the literature by comparing the relative influences of consumer word-of-mouth with reviews from professional critics. To empirically estimate and separate the effects of the two signals, we analyze a sample of 1,480 video games and their sales figures between 2004 and 2010. Such a comparison is potentially valuable given the prevailing view that word-of-mouth and other content generated by users is becoming increasingly influential in consumer decision-making, possibly even to the exclusion of traditional reliance on the opinions of experts or professional critics. Our analysis considers the possibility that these signals affect consumer behavior jointly and separately through a more detailed examination of interactions between signals and the subsequent effect on sales performance.

The findings of this study reinforce the hypothesis that the reviews of professional critics associate strongly with commercial success. After taking steps to control for endogeneity using a generalized method of moments (GMM) estimator, we find evidence that reviews from professional critics influence sales instead of merely predicting them, suggesting that their independence and reputation serve as a credible signal that helps consumers support the decision-making process by minimizing uncertainty. We also find only limited evidence to suggest that the valence of consumer word-of-mouth affects product sales once we

control reviews from professional critics and interaction terms. Consequently, our results are contradictory to a commonly held belief in the value of consumer word-of-mouth and emphasize the greater importance of reviews from professional critics in the digital market context.

Regarding placing the study and findings within the context of the existing literature, the results reinforce some well-established findings while presenting alternative and some-times contradictory evidence on others. We arrive at contradictory results in comparison with Chintagunta et al. [CGV10], given that they highlight the importance of the valence of word-of-mouth and find neither evidence for the significant explanatory power of reviews from professional critics nor the volume and valence of user reviews. We essentially arrive at the opposite conclusion, potentially due to our aggregation to the national level versus their regional-level analysis, an acknowledgment the authors make in their paper. Instead, our findings highlight the importance of simultaneously controlling for reviews from critics and consumer word-of-mouth.

Consequently, consumers likely assess the credibility and reliability of the interaction of distinct types and similar types of signals jointly. Although there is a shortage of studies in marketing on the interaction of signals, the economics literature has produced several studies on the value of multiple signals of product quality, most notably concerning the moral hazard associated with agency theory [Hol79] and most often considered concerning issues of corporate governance and performance-related pay [FS11]. However, the effect of additional signals may diminish at the margin as the total number of available signals increases. Basuroy et al. [BDT06] are among the first to empirically study the interaction between quality signals. Other authors, such as Kirmani and Rao [KR00], also account for the interaction between a limited number of independent types of signals in their theoreti-cal framework. Following the studies and principles of signaling theory, we consider the importance of additional signals and their interactions.

Frick and Kaimann [FK17] have used real-world data to study the impact of customer reviews on market demand in electronic markets for mobile applications. Using data from the Apple App Store, we analyzed a sample of 32 applications with 5,792 daily observations and their number of installations during the first six months of 2015. The applications were randomly selected from each category in the App Store. The findings extend the current literature, which has only partially examined the influence of ratings and ad expenditures separately and jointly on downloads. The results show a positive interaction between valence and variance. In addition, the interaction between valence and variance has a more significant positive effect on quality perceptions and, thus, a more considerable impact on application downloads.

Furthermore, the empirical findings also support that customer reviews and marketing efforts boost installations. However, if they co-occur, the influence of both effects will be diminished. Thus, our findings advance reputation analyses by explicitly considering the possibility that these signals affect consumer behavior jointly and separately by conducting a more detailed analysis of interactions in electronic markets.

Effect of different sources of online review variance on product prices and demand at the

level of firms

Consumer ratings enable prospective consumers to learn from other consumers' experiences. This means that the reporting of experiences within an online rating enables new consumers to assess experience attributes of a product or service (i.e., those attributes that can hardly be observed prior to purchase but only after the purchase) better than before the emergence of consumer ratings. Thus, consumer ratings transform the product or service's experience attributes into attributes that can be searched within the reviews of this product or service. For instance, by reading online reviews, potential customers can learn about past customers' experiences concerning the ease of navigating over an application's menu. Some customers might like a simple menu, whereas others might prefer a more complex menu with highly adjustable settings. Customers' disagreements resulting from opposing opinions are thus caused by *taste differences* and might not necessarily imply a bad product or service for all consumers. However, potential customers are able to learn not only about the different taste-related experience attributes associated with an application but also about (potential) service failures, i.e., *quality differences*. For instance, if an application fails to work on a specific data set, consumers are also likely to report this failure within their online review. Both types of experience attributes (taste-related and quality-related) are likely to induce additional variance of a product. The key difference between both sources of variance is that all potential customers would agree about their dislike of variance caused by *quality differences* but not necessarily by the variance caused by *taste differences*.

To examine the relationship between the variance caused by *taste* or *quality differences* and a product's or service's prices and demand, the authors develop a two-period analytical model featuring a monopoly retailer and consumers that differ in taste and risk aversion. The first period explains the review-generation process, whereas, in the second period, the effects of how new consumers use the generated reviews from the first period in their decision-making process are examined. In the first period, where the product or service has no online reviews yet, a set of innovators enter the market. The retailer sets a profit-maximizing price based on its expectations about the product's or service's characteristics as well as the expected utility for the consumer. Afterward, the innovators decide whether to purchase the product or service based on the price and their expectations about the characteristics of the product or service. Then, the purchasing innovators (or at least those with extreme experiences) publish honest ratings about the product. In the second period, where online reviews for the product or service are present from the first period, a set of imitators enter the market. The retailer observes the consumer ratings of innovators and sets a profit-maximizing price for the product based on the observed consumer ratings. Afterward, imitators observe the consumer ratings and derive the product's or service's characteristics through these reviews. This means that imitators have no remaining uncertainty about the experience attributes of the product or service. Next, they decide whether or not to purchase the product or service based on the price and the observed consumer ratings. The authors analyze two types of goods: (1) consistent quality goods, where the variance of consumer ratings is solely caused by *taste differences* (model based on Sun [Sun12] to connect with prior literature), and (2) inconsistent quality goods, where the variance of consumer ratings is caused by *taste differences* and *quality differences* in the form of product failure.

Concerning the case of consistent quality goods, the theoretical model predicts that price

and demand both increase with the average rating, as a higher average rating is a credible signal of high product quality. This result represents a theoretical confirmation of prior empirical findings [LH08]. With an increasing variance of ratings (which is solely caused by *taste differences*), the price increases while the demand for the good or service decreases. The intuition behind this relationship is as follows: An imitator with a taste that closely matches the product or service enjoys this product or service more than a product with a low variance of ratings. The retailer charges a higher price to all imitators to skim the higher willingness to pay of imitators with tastes that closely match the product or service. This higher price deters imitators with tastes that do not closely match the product or service, resulting in lower demand. Figure 6 illustrates the relationship between the products or services online rating variance and the price and demand of the product's or service's, respectively.
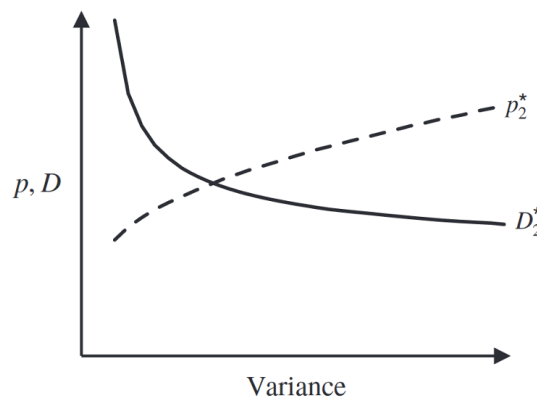

Figure 7: *Optimal price and demand for consistent quality goods. (Source: Own illustration)*

Note: For reasons of simplicity, the price and demand are plotted on the same axis. p_2^* represents the optimal price for the product or service in the second period, and D_2^* represents the optimal demand in the second period.

Regarding the analysis of a product or service that can be characterized as an inconsistent quality good (i.e., a product or service that can fail), the model differs in two major ways. First, the model now also takes the consumer's individual risk aversion (i.e., the negative utility caused by the risk that the product or service fails) into account, which in turn is associated with the consumer's purchase intentions. Second, if a consumer in the first period buys a product or service that failed, the consumer will publish the lowest rating possible. In contrast to consistent quality goods, the expected enjoyment of inconsistent quality goods depends not only on two but on three product characteristics: the average rating, the variance caused by *taste differences*, and the variance caused by *quality differences*. Similar to the case of consistent quality goods, the model predicts that price and demand will both increase with the average rating of an inconsistent quality good, as a higher rating acts as a credible signal of high product quality. Further, if the variance caused by taste differences increases, the price increases, and the demand decreases for an inconsistent quality good. The intuition behind this result is the same as in the consistent quality goods case. However, for inconsistent quality goods, if the variance caused by *quality differences* increases, the price of the product or service will decrease. This is

because a higher variance caused by quality differences indicates a high failure rate of the product or service, which represents a signal for low quality and will thus reduce the price.

To examine the effects on the demand for the product or service, there are two distinct effects that need to be taken into consideration: (1) the price effect, which states that a reduced price will increase the demand, and (2) the failure effect, which models the relationship that a higher probability of product failure will decrease the demand. If the variance caused by *quality differences* is sufficiently low and the variance caused by *quality differences* increases, then the model predicts that the demand effect will be smaller than the failure effect, resulting in a decreased demand. A somewhat counterintuitive case occurs if the variance caused by *quality differences* is sufficiently high and the variance caused by *taste differences* is sufficiently low. Then, the price effect will be greater than the failure effect, meaning that the demand will increase with an increasing variance caused by *quality differences*.

Bearing these results in mind, if the total variance is constant and the decomposition of the source of variances changes to a higher relative share of variance caused by *taste differences*, then the price will increase. This is because the probability of failure due to *quality differences* will become smaller, and the retailer will have more power to raise the product's or service's prices. If the total variance is sufficiently low, then demand increases with an increasing share of variance caused by *taste differences* (see Figure 7, left side). On the contrary, if the total variance is sufficiently high, the demand will decrease with an increasing share of variance caused by *taste differences* (see Figure 7, right side).
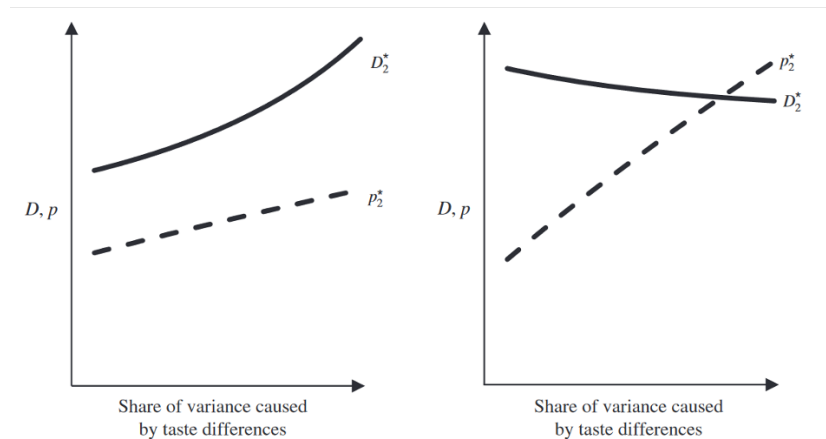
Figure 8: *Optimal price and demand for inconsistent quality goods—Changes in the composition of the variance. (Source: Own illustration)*

Note: For reasons of simplicity, the price and demand are plotted on the same axis. The left graph depicts the case where the total variance is sufficiently low, whereas the right graph depicts the case where the total variance is sufficiently high. p_2^* represents the optimal price for the product or service in the second period, and D_2^* represents the optimal demand in the second period.

The results of the paper have important managerial implications for the development of an OTF market. If service providers on OTF markets were to consider the composition of the

variance of consumer ratings, then they could improve their sales forecasts and increase profits by adjusting their inventories accordingly to satisfy demand or by charging higher prices for those products or services for which a relatively larger share of the variance is caused by *taste differences*. Additionally, service providers on OTF markets could implement mechanisms to explicitly communicate information about the decomposition of the variance to allow more consumers to use this important information in their decision-making.

Selected propositions from the analytical model of this research were also empirically confirmed within Subproject A4 [Gut+18]. Within this study, the author employs a machine learning approach to decompose the variance caused by *quality differences* (i.e., product failure) and *taste differences* for digital cameras on Amazon.com and estimates its impact on the product's price and demand. In line with the predictions of the theoretical model, the author finds that variance caused by *quality differences* is negatively associated with price and demand and that the variance caused by *taste differences* positively affects the product's price and demand. Moreover, the theoretical insights by Zimmermann et al. [ZHKN18] have also informed and inspired marketing research [LBS22] to further extend the theoretical model developed within this paper and establish cumulative knowledge on this topic.

Impact of local market competition on the heterogeneity of available businesses on the market level
However, an unresolved question on the more aggregated market level remained. For instance, it has remains unclear what impact the local market competition has on the heterogeneity of available businesses in a market. Even though prior work has investigated the formation of mean ratings for businesses or the nature of particular reviews [LH08], these studies do not inform about the relationship between competition in a local market and the properties of a market's mean rating distribution. When mean rating distributions change with competition, this means that a business's mean value has to be evaluated differently in markets with the different competition. The relative position in the market of a 3.5-star business will be different in a market of low competition compared to a market with high competition. Therefore, it is crucial to better understand that ratings must be considered within a bigger picture, i.e., taking into consideration boundary conditions in the process of decision-making. The article by Gutt et al. [GHR19] contributes to this notion and provides empirical evidence for this presumption.

While the theoretical literature in industrial organizations has analyzed equilibrium market outcomes for vertically differentiated industries where quality provisioning is primarily tied to marginal costs (e.g., restaurants), the empirical evidence was still scant. Theory suggests that larger markets can support a greater number of firms that cover a larger quality spectrum than smaller markets, such that both tails of the distribution of available qualities grow. So far, the empirical evidence was limited to the growth in the higher end of the quality distribution [BW10]. By studying how local market competition affects the dispersion in both tails of a market's mean rating distribution, Gutt et al. [GHR19] were able to also investigate the dynamics in the lower end of the service quality distribution and evaluate the impact on the average of the distribution.

What we see as an important lesson learned from this work is that the range and av-

erage of a market's mean rating distribution may vary depending on the competition level in the particular markets. Here, the difference between the very best and very worst of individual mean ratings in a particular market is described by the range, and the average consists of all individual mean rating valences within that market. To compare different markets, Gutt et al. [GHR19] identified markets (i.e., cities) that do not overlap (i.e., markets which are isolated). The competition level within a market is determined by the total number of businesses. Based on a comprehensive data analysis on a combined data set from *Yelp.com* and *city-data.com*, the authors found that the range of markets' mean rating distributions increases with competition (i.e., number of businesses) but that the average of the markets' mean distributions decreases (see Figure 8). This means a 4.5-star business competes with more 4.5-star businesses in a small market than in a larger market (arrow (a)), whereas a 2.5-star business competes with relatively fewer comparable businesses—in terms of mean ratings-in a small market than in a large one (arrows (b)). With regard to the range of the mean rating distribution, a 4.5-star rating business might be among the upper businesses in a small market but might be considered less elite, being in a larger market due to the wider range on large markets (arrow (c)). While a 2.5-star business might be among the worst businesses in a small market, the worst choices in larger markets have even lower mean ratings (arrow (d)).
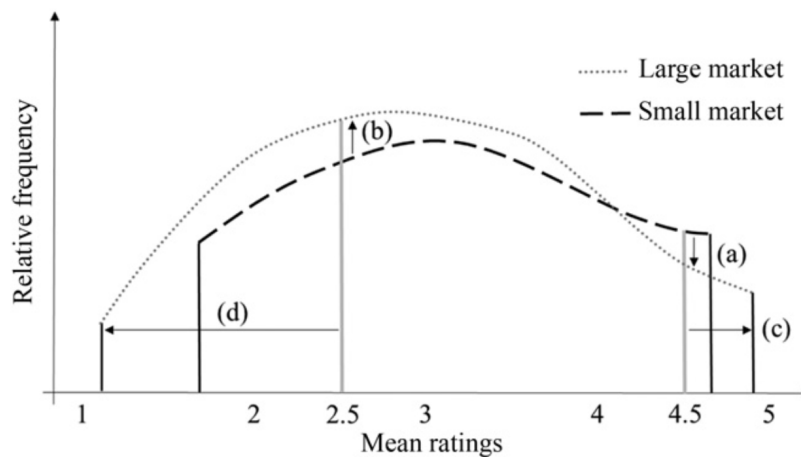
Figure 9: *Competition faced by 2.5- and 4.5-star businesses in two different markets.*

By conducting multiple regression estimations (e.g., cross-sectional estimation, ordinary least squares, and two-stage least square, including instrumental variables, fixed and random effects) on the combined data set from Yelp.com and city-data.com, it was possible to show that local market competition is a driver of the heterogeneity of available mean ratings in a market [GHR19].

One conclusion from these results is that a larger market has proportionately more lower-rated businesses. In contrast, higher-rated businesses have relatively fewer comparable businesses and face less competition in such a market. As market size increases, firms proliferate such that many different quality levels become available, assuming that ratings reflect the quality levels approximately. In these markets, firms compete on the quality of the service they sell through a combination of fixed and variable costs. A larger market can thus support more low-quality firms and firms that offer higher qualities than the

highest-quality firm in a small market.

Consequently, as competition increases, so does the range of different qualities available in the market. The change in the average quality of the market level depends on the ratio of low- and high-quality firms entering the market. In other words, it decreases if the increase in competition is mainly driven by low-quality rather than by high-quality firms. Previous empirical studies from industrial organizations have documented that increasing competition leads to an increase in the dispersion in the higher end of the service quality distribution [BW10]. Yet, the effect on the dispersion in the lower end of the distribution has been neglected. Gutt et al. [GHR19] extend this result by finding that the dispersion in the lower end of the service quality distribution exceeds that of the higher end. Moreover, literature on business intelligence and analytics (as suggested by Chen et al. [CCS12]) can build on this evidence—in particular, Yelp's mean ratings form an internally consistent data source for conducting competitive intelligence activities.

Furthermore, the insights of this work lead to the implication for requesters interested in assessing applications with the same functionality (e.g., navigation via online maps) based on electronic word-of-mouth, such as ratings. This means that applications across OTF markets with similar mean ratings should be assessed differently by considering the competition level. Different actors such as requesters (i.e., an entity requesting the creation of an application consisting of components), component providers (i.e., an entity offering access to a component), or infrastructure providers (i.e., an entity offering access to infrastructure for the execution of services) participate in OTF markets. These entities might only have access to a particular set of OTF markets due to restrictions or strategic decisions (e.g., internet censorship or ensuring independence). Moreover, the market provider (i.e., an entity operating and providing access to an OTF market) might decide to restrict access for certain actors. For example, Apple or Google can be seen as a market provider in the case of the App Store and Google Play, where they can decide who is allowed to provide apps and install them. Thus, OTF markets can be seen as isolated markets, for example, geographically or concerning an application domain that is specifically separated. Thus, some entities might exist that act across different OTF markets. Therefore, OTF markets can have different sizes regarding the number of component or infrastructure providers. Requesters might compare applications (i.e., compositions of components) providing this functionality across different OTF markets [KKMW20]. In these cases, the rating distributions for the applications might change with competition within individual OTF markets due to the competition between component providers. Thus, requesters might want to evaluate the mean values of applications differently between OTF markets with different levels of competition. In general, requesters who prefer applications with very high mean ratings will find more suitable matches in larger OTF markets. However, requesters should not randomly choose an application because larger markets are home to disproportionally more applications with low mean ratings than smaller markets. In other words, the broader range of available mean ratings comes at the expense of lower averages of the mean rating distributions in larger markets. This result emphasizes the importance of online review systems and other mechanisms to prescreen applications in large OTF markets to serve requester preferences better.

2.2   Opportunistic Behavior of Service Providers

Strategic decisions of service providers when customer feedback is inaccurate
Mir Djawadi, Fahr, Haake, and Recker [MFHR18] have focused on reputation systems
with inaccurate customer feedback. Since customers may not have the expertise to judge
the respective service quality correctly, customer feedback in reputation systems is not
always accurate. Consequently, sellers may engage in strategic behavior in quality choices,
i.e., they may deliver low quality after having achieved a high sales price due to sufficiently
many positive ratings.

The research strategy involves a theoretical model and a laboratory experiment. Both
components are the result of a collaboration between the Subprojects A3 and A4. In the
theoretical analysis, a service provider repeatedly sells a service to short-lived customers,
i.e., customers interact only once with the service provider). The rating behavior of the
customer is modeled as a random variable, and the service provider's strategic quality
choice is modeled as a stochastic optimization problem using a discounted Markovian
decision process. The service provider chooses to produce either high- or low-quality
service (at high and low costs, respectively) and receives a sales price depending on the
current reputation profile. Profit changes are modeled based on reputation profiles by
keeping the demand constant for all service providers and changing the price a single
service provider may charge. After purchase, a customer either positively or negatively
evaluates the service according to predefined error probabilities, reflecting the customer's
limited ability to judge the provided service quality correctly.

The objective of the laboratory experiment is to investigate how subjects in the role
of service providers make their quality choices and compare observed with optimal be-
havior derived from the theoretical model. In the experiment, all student subjects play
the service provider role and choose between producing a high- or low-quality service
(associated with high and low costs, respectively). Four treatments are implemented with
exogenous variation in the customers' evaluation abilities to rate the service quality accu-
rately. All other parameters of the theoretical model are held constant. The consideration
of exogenous rating behavior depending on different evaluation abilities of the customer
and a sales price resulting directly from the reputation profile allows for identifying the
reputation system's causal effects on the service provider's quality choice.

Theoretically, a profit-maximizing service provider should always produce high-quality
or low-quality services, irrespective of the current quality profile. This means that a good
reputation is maintained optimally, or any reputation-building process is wholly neglected.
However, according to the results of the experiment, not all subjects follow the optimal
strategy in the treatment. Instead, the higher the propensity to choose high quality, the more
accurately customers can rate the quality. Moreover, the chosen qualities are conditional
on the current reputation profile. More precisely, many subjects use milking strategies and
exploit a good reputation. Thus, low quality is delivered if the sales price is high until the
price drops below a certain threshold due to negative ratings. High quality is chosen until
the price has significantly increased.

2.3 Design of Online Review Systems in OTF Markets

How customers aggregate information to assess product quality
Van Straaten, Melnikov, Hüllermeier, Mir Djawadi, and Fahr [SMH+21] have focused on aggregation patterns of customers on selling platforms. Selling platforms usually provide aggregated measurements of customer feedback in which the numerical part of customer reviews is processed to single index values representing the valence of the product, most often implemented by the arithmetic mean. However, literature in psychology and behavioral economics has identified behavioral biases driven, e.g., by bounded rationality or heuristics (e.g., [TK74]). Therefore, it is unclear whether the technically implemented aggregation functions represent the actual aggregation behavior of customers. Van Straaten et al. [SMH+21] conduct a laboratory experiment to elicit subjects' aggregation patterns and compare these to reference aggregation functions. Thus, they investigate to which degree inherent heuristics of customers affect the aggregation of customer rating distributions and whether they result in systematic biases that should be addressed in the implemented aggregation metrics in reputation systems.

In the experiment, student subjects rank various customer rating distributions to infer the subjects' aggregation heuristics. This allows for finding the optimal aggregation function and comparing it with different alternative aggregation functions. More precisely, subjects receive customer ratings of three products and are asked to rank them according to their preferences. These customer ratings differ concerning their relative frequencies and their arithmetic means. Subjects only see the products' aggregated customer ratings and know that they are from the same product category and have similar prices and specifications, but not the products' names or detailed specifications. Aggregated ratings that allow separating rankings based on different decision heuristics are used, such as minimization of negative ratings or maximization of positive ratings from rankings that favor the arithmetic mean. Subjects choose rankings for a total of 12 categories, whereby distributions are partly artificial and partly based on aggregated customer ratings from the Amazon marketplace. To elicit actual preferences, the entire ranking decision is incentivized: subjects receive a USB flash drive that they rank first or second as payment and have the chance to win another product they choose from one of two other product categories (a tablet computer or tablet holder). They have a higher chance to win the products when they rank a product better and do not know which decision determines their payoff. We implement two treatments investigating the impact of additional numerical information on aggregation heuristics. Subjects are only given the aggregated customer ratings in the control treatment without additional information. In contrast, in the information treatment, subjects also see the relative frequency of each rating category and the arithmetic mean value associated with the distribution. To analyze subjects' rankings, the authors employ a Maximum-Likelihood approach and a Placket-Luce model, which is a model of rank data that is parametrized by quantitative preference degrees for individual choice alternatives.

The results of the experiment show that the arithmetic mean is an appropriate aggregation function as it best explains subjects' average behavior. However, the findings also reveal a tendency to overweight moderate ratings and underweight extreme one, thus indicating a binary bias that is not sufficiently considered in practice. Moreover, minor subject clusters focus on customer rating distributions with the least 1-star ratings or the

least negative (i.e., 1- and 2-star) ratings. Contrary to predictions, individual characteristics, risk attitudes, online shopping experience, and additional numerical information do not affect the employed aggregation heuristics. As the findings indicate heterogeneity in aggregation behavior, marketplaces could enhance reputation systems by calculating personalized valence values.

Influence of anonymity and self-expression on the propensity of writing online reviews
Hoyer and van Straaten [HS22] have dealt with anonymity and self-expression in online rating systems. Since providing a customer review costs time and effort without an obvious benefit for the reviewer, drivers of writing a review have been analyzed frequently in the literature. Studies show that the main motives for writing customer reviews are self-expression, altruism, a sense of belonging, and economic incentives (e.g., [CL12]). On the other hand, sharing experiences about products and services might be accompanied by concerns about sharing private information and the threat of being involuntarily analyzed and clustered by third parties. To avoid this privacy issue, anonymous reputation systems can be implemented. However, this anonymity can potentially crowd out reviews that are motivated by the self-expression motive of the customers. To date, the impact of anonymity and its driving forces (such as the blunting of self-expression) on the propensity of customers to write reviews is unclear [RS12]. Therefore, Hoyer and van Straaten [HS22] conduct an experimental study to answer the following research question: Do anonymous reputation systems have a drawback of blunting self-expression as a motive of customer reviews and do altruistic attitudes moderate this effect?

The laboratory experiment consists of a stylized marketplace in which all participants are customers facing computerized sellers. The participants must decide which seller to buy a product from in 10 separate periods. After choosing a seller and learning how satisfied they are with the product, they are given the choice of whether they want to publish their satisfaction with the product, which will be visible to all participants. The two treatments enable to vary the degree of anonymity–from pseudonymity (chosen pseudonym, published when satisfaction is revealed) to anonymity (assigned buyer ID, not published when satisfaction is revealed)–and analyze whether this treatment influences the self-expression motive and hence the propensity of customers to leave reviews. Addressing the self-expression motive, in the pseudonymity treatment, a ranking of the participants' number of published ratings is shown at the end of each period, enabling them to build up a reputation as frequent reviewers. This is not the case in the anonymity treatment, as ratings are anonymous. After the market experiment, a dictator game analyzes participants' attitudes toward altruism. Conducting an economic experiment allows drawing inferences about the effect of anonymity on self-expression with high internal validity as it is controlled for other elements such as prices, product groups, visualization effects, etc.

The results indicate that self-expression is indeed blunted by anonymity. This result is driven by less-altruistic subjects, as more-altruistic subjects are not affected by the blunting of self-expression. In line with the literature on signaling theory, in the experimental setup with a focus on intrinsic motivation and self-expression, excluding the self-expression motive indeed decreases welfare in terms of monetary payoffs. The study thus carries substantial implications, as it identifies the necessity to consider the drawbacks of anonymous reputation systems that must be considered and balanced with the advantages of anonymity

(i.e., reduced privacy issues), which might depend on the actual market environment and the products traded on these markets

Connection between social proximity among trading partners and upward-biased ratings
Mir Djawadi and Wester (2022) [MW22] have examined to what extent social proximity between trading partners leads to upward-biased ratings. Peer-to-peer platforms placed under the 'sharing economy', such as Airbnb or Couchsurfing, Uber, or BlaBlaCar, have recently stirred the electronic commerce landscape. Thereby, social interaction between customers and service providers is not only an inherent part, but many people participate for this reason. Due to the intimate nature of the arrangements offered (e.g., homestay or car share), sharing-economy markets depend on trust. Where considerable heterogeneity in service quality and information asymmetries among geographically dispersed and informally connected users exist, credible reputation-based feedback systems become essential to support the emergence of trust. However, many reputation systems have given rise to implausible rating distributions in such a way that ratings cannot be assumed to reflect the actual quality, calling the functionality and effectiveness of reputation systems based on user-generated feedback into question. An examination of the literature suggests diverse reasons for overwhelmingly skewed ratings, among them being non- or under-reporting of negative experiences due to fear of retaliation or social interactions between market participants (e.g., [DW08]). While there are already several suggestions for the design of reputation systems to counteract upward-biased reviews due to fear of retaliation, examining the influence of social interaction on feedback giving has only recently gained momentum.

Through the exchange of personal information, familiarity and liking between individuals are established, uncertainty about others' intents can be reduced, and generalized positive beliefs or impressions may be evoked. Knowledge about one another is found to be linked to a multitude of pro-social behaviors. A well-documented and robust finding is that interpersonal similarity has a reinforcing effect. People perceived to be like themselves are more likely to be regarded as more familiar and socially proximate. Along this line, recent research consistently finds that people are more willing to help, more lenient toward misbehavior, and trust in and feel morally obliged to interact with partners who are socially proximate (e.g., [BWD14]). Common explanations suggest that people derive greater personal satisfaction or utility from helping socially proximate individuals and/or have generalized expectations that those being similar are also more likely to be well-disposed toward them. Hence, Mir Djawadi and Wester (2022) [MW22] propose that feedback giving differs depending on the availability of personal information and social proximity between transaction partners. It is expected that disclosing personal information per se leads to better feedback than having no information at hand. Second, it is hypothesized that socially proximate interaction partners receive better feedback than socially distant interaction partners.

Given the lack of either directly observing social interaction (frequency and intensity) or assessing the degree of social proximity between transaction partners, laboratory experiments can help complement observational data derived from peer-to-peer platforms. Experiments allow the comparison of individuals' feedback-giving behavior with varying

degrees of social proximity between transaction partners potentially arising from personal interactions. Therefore, the experiment simulates a sharing economy market framework where a provider offers service and is remunerated and evaluated by a customer taking up the offered service. Since a crucial feature of the sharing economy is the potential for collective action taken by both sides to enhance the experience, the quality of the service is modeled as a function of the joint effort exerted by providers and customers. Whereas these elements are static across treatments, the extent of personal information presented to transaction partners varies.

2.4 Certification to Reveal Service Quality in OTF Markets

The impact of self-declaration and certification on price premiums for experience goods
Fanasch and Frick [FF20] have used the wine industry as an example, as many producers have adopted organic or biodynamic certifications to reveal their production quality. In principle, certifications can be granted for organic and biodynamic practices. However, the few available studies either concentrate on organic practices or consider organic and biodynamic production jointly [AGG17]. However, it is essential to investigate the effect of each practice separately, as organic production is usually considered "serious." In contrast, biodynamic production is often considered "bizarre" or "somewhat strange," partly due to the methods used.

While some of these wineries have successfully applied for third-party certification, others follow strict guidelines without being certified and self-declare themselves to be eco-friendly. Using a large sample of 55,500 wines produced by 1,514 German wineries between 2010 and 2017, this study estimates a series of hedonic price models across different price quantiles. The results indicate a statistically significant price premium for organic and biodynamic certified experience goods, the magnitude of which is, however, far smaller than the effects usually identified in surveys and laboratory experiments. While self-declaration is only a credible signal for organic practices generating a price premium of 8.6 percent, biodynamic practices require certification for a price premium of 4.1 percent. The results also suggest that the interaction of collective reputation and biodynamic practices positively impacts prices.

Overall, this study contributes to current research by closing several research gaps. First, this study compares the effects of self-declaration and certification on product prices by distinguishing between organic and biodynamic practices. Second, this study conjectures that the impact of self-declaration and certification is likely to differ across bottle price distribution, as Abraben et al. [AGG17] suggested. This provides a more detailed understanding of the relationship between prices and self-declared and certified sustainability practices. Third, this paper provides new evidence on whether and to what extent collective reputation impacts the prices that organic and biodynamic wineries can charge. Given the recent changes in consumer preferences, this study uses the latest data available to identify customers' willingness to pay for organic and biodynamic wines.

Thus, this study contributes to information asymmetry and signaling theory by showing that certification costs can be avoided since, in some instances, self-declaration is

sufficient to enable a price premium. As a result, certifications are essential for decreasing information asymmetries in markets for composed services. By receiving an external certification, providers can reliably indicate that they meet particular (minimum) standards.

3   Concluding Remarks

The empirical research of Subproject A4 has significantly contributed to the area of online reviews and certification in reducing information asymmetries in OTF markets.

By using econometric analysis of existing markets with comparable characteristics of the OTF market, various solution concepts to signal true service or product quality have been thoroughly investigated. Beyond the benefits of regular customer ratings, research by Cox and Kaimann [CK15] has shown that ratings from professionals or experts can even enhance the customer's confidence in the quality of a service/product which, in turn, can increase demand and reduce market inefficiencies. OTF market providers might therefore consider allowing different types of signals in the reputation system to help customers make the best selection among the offered services. Further research addressed the question whether the seemingly strongest solution concept of certification should be part of a reputation system as well. As Fanasch and Frick [FF20] have shown, there are costless substitutes for certification. Especially if service providers already built a high reputation, certifying their services or processes might only incur unnecessary costs. As costless self-declaration can lead to comparable levels of trust, services can be offered at a lower price which, would not only benefit current customers but may also attract new customers on the respective OTF market.

With regard to online reviews and economic outcomes, Zimmermann et al. [ZHKN18] were able to investigate analytically how different sources of variance of online reviews affect product prices and demand on the level of firms. It led to further research, such as Gutt [Gut+18], who empirically confirmed selected propositions from the analytical model provided by Zimmermann et al. [ZHKN18], or Lee et al. [LBS22], who extended the model and thereby introduced it into marketing research. An implication of Zimmermann et al. [ZHKN18] for service providers on OTF markets is that if they consider the composition of the variance of consumer ratings, then these providers could improve their sales forecasts and increase profits by adjusting their inventories accordingly to satisfy demand or by charging higher prices for those products or services for which a relatively larger share of the variance is caused by taste differences. Additionally, service providers on OTF markets could implement mechanisms to explicitly communicate information about the decomposition of the variance to allow more consumers to use this important information in their decision-making.

The interplay between the design of the market infrastructure (regarding the possibilities of quality assurance) and individual decisions (supply, demand, price, and quality of the traded service) has been systematically investigated. In this context, Gutt et al. [GHR19] were able to investigate empirically what impact the local market competition has on the heterogeneity of available businesses on a market level. Gutt et al. [GHR19] also provided input for further research in highly reputable journals such as the Information

Systems Research (e.g., [AR22]). In the OTF context, an implication of the findings made by Gutt et al. [GHR19] is that OTF market providers might want to consider the competition level within the review system for requesters interested in assessing applications with the same functionality across OTF markets. As those applications with similar mean ratings should be assessed differently by considering the competition level, the market providers might want to add an overview similar to Figure 8, where the mean ratings and their distribution for different OTF markets can be observed or corrected directly by providing adjusted measures depending on the competition levels when comparing between markets.

With the help of economic experiments that have been created in studies like Mir Djawadi et al. [MFHR18], causal for example, analysis of designs of reputation systems or measures (certificates, contract structures, etc.) was systematically conducted that would be otherwise difficult to achieve with field data–especially when features that do not yet exist and have no comparable counterpart in existing markets have to be considered. The results from these experiments have been valuable for revealing behavioral patterns that partly deviate from traditional economic theory and the assumption of rational actors who are only interested in maximizing individual payoffs. For example, the research revealed that a substantial share of service providers milk their good reputation, which represents a non-rational oscillating strategy, an economically irrelevant factor of social proximity seems to influence the rating behavior of market participants, and customers tend to overweight moderate and underweight extreme ratings. This excerpt of empirical findings suggest that theories of Behavioral Economics that relax the assumptions of perfect rationality and selfishness should be considered as alternatives for explaining behavior on OTF markets and designing interventions that prevent market failure. Further, the results of experiments on the acceptance of anonymous online rating systems can be used to develop rating systems that holistically reflect product or service quality and can thus serve to reduce information asymmetries. For example, research on the metrics of online valuations can be used to embed the aggregated valuations intentionally. These suitable rating systems ensure the necessary security and acceptance and, thus, the use of the rating systems. These results can be used in the conceptual development of rating systems as an element of the business model of a market provider in OTF markets.

Bibliography

[AGG17]   ABRABEN, L. A.; GROGAN, K. A.; GAO, Z.: Organic price premium or penalty? A comparative market analysis of organic wines from Tuscany. In: *Food policy* 69 (2017), pp. 154–165

[ake78]   AKERLOF, G. A.: The market for "lemons": Quality uncertainty and the market mechanism. In: *Uncertainty in economics*. Elsevier, 1978, pp. 235–251

[AR22]   ALYAKOOB, M.; RAHMAN, M. S.: Shared prosperity (or lack thereof) in the sharing economy. In: *Information Systems Research* (2022)

[BDT06]   BASUROY, S.; DESAI, K. K.; TALUKDAR, D.: An empirical investigation of signaling in the motion picture industry. In: *Journal of marketing research* 43 (2006), no. 2, pp. 287–295

[BW10]   BERRY, S.; WALDFOGEL, J.: Product quality and market size. In: *The Journal of Industrial Economics* 58 (2010), no. 1, pp. 1–31

[BWD14]   BALLIET, D.; WU, J.; DE DREU, C. K.: Ingroup favoritism in cooperation: a meta-analysis. In: *Psychological bulletin* 140 (2014), no. 6, p. 1556

[CCS12]   CHEN, H.; CHIANG, R. H.; STOREY, V. C.: Business intelligence and analytics: From big data to big impact. In: *MIS quarterly* (2012), pp. 1165–1188

[CGV10]   CHINTAGUNTA, P. K.; GOPINATH, S.; VENKATARAMAN, S.: The effects of online user reviews on movie box office performance: Accounting for sequential rollout and aggregation across local markets. In: *Marketing science* 29 (2010), no. 5, pp. 944–957

[CK15]   COX, J.; KAIMANN, D.: How do reviews from professional critics interact with other signals of product quality? Evidence from the video game industry. In: *Journal of Consumer Behaviour* 14 (2015), no. 6, pp. 366–377

[CL12]   CHEUNG, C. M.; LEE, M. K.: What drives consumers to spread electronic word of mouth in online consumer-opinion platforms. In: *Decision support systems* 53 (2012), no. 1, pp. 218–225

[DW08]   DELLAROCAS, C.; WOOD, C. A.: The sound of silence in online feedback: Estimating trading risks in the presence of reporting bias. In: *Management science* 54 (2008), no. 3, pp. 460–476

[FF20]   FANASCH, P.; FRICK, B.: The value of signals: Do self-declaration and certification generate price premiums for organic and biodynamic wines? In: *Journal of cleaner production* 249 (2020), p. 119415

[FK17]   FRICK, B.; KAIMANN, D.: The impact of customer reviews and advertisement efforts on the performance of experience goods in electronic markets. In: *Applied Economics Letters* 24 (2017), no. 17, pp. 1237–1240

[FS11]   FAHLENBRACH, R.; STULZ, R. M.: Bank CEO incentives and the credit crisis. In: *Journal of financial economics* 99 (2011), no. 1, pp. 11–26

[GHR19]   GUTT, D.; HERRMANN, P.; RAHMAN, M. S.: Crowd-driven competitive intelligence: Understanding the relationship between local market competition and online rating distributions. In: *Information Systems Research* 30 (2019), no. 3, pp. 980–994

[Gut⁺18]   GUTT, D. et al.: *In the Eye of the Beholder? Empirically Decomposing Different Economic Implications of the Online Rating Variance*. Tech. rep. Paderborn University, Faculty of Business Administration and Economics, 2018

[HMM12]   HENNIG-THURAU, T.; MARCHAND, A.; MARX, P.: Can automated group recommender systems help consumers make better choices? In: *Journal of Marketing* 76 (2012), no. 5, pp. 89–109

[Hol79]   HOLMSTRÖM, B.: Moral hazard and observability. In: *The Bell journal of economics* (1979), pp. 74–91

[HS22]   HOYER, B.; STRAATEN, D. van: Anonymity and self-expression in online rating systems—An experimental analysis. In: *Journal of Behavioral and Experimental Economics* 98 (2022), p. 101869

[KKMW20]   KARL, H.; KUNDISCH, D.; MEYER AUF DER HEIDE, F.; WEHRHEIM, H.: A case for a new IT ecosystem: On-The-Fly computing. In: *Business & Information Systems Engineering* 62 (2020), no. 6, pp. 467–481

[KR00]   KIRMANI, A.; RAO, A. R.: No pain, no gain: A critical review of the literature on signaling unobservable product quality. In: *Journal of marketing* 64 (2000), no. 2, pp. 66–79

[LBS22]   LEE, N.; BOLLINGER, B.; STAELIN, R.: Vertical Versus Horizontal Variance in Online Reviews and Their Impact on Demand. In: *Journal of Marketing Research (JMR)* (2022)

[LH08]   LI, X.; HITT, L. M.: Self-selection and information role of online product reviews. In: *Information Systems Research* 19 (2008), no. 4, pp. 456–474

[MDC14]   MAYZLIN, D.; DOVER, Y.; CHEVALIER, J.: Promotional reviews: An empirical investigation of online review manipulation. In: *American Economic Review* 104 (2014), no. 8, pp. 2421–55

[MFHR18]   MIR DJAWADI, B.; FAHR, R.; HAAKE, C.-J.; RECKER, S.: Maintaining vs. milking good reputation when customer feedback is inaccurate. In: *Plos one* 13 (2018), no. 11, e0207172

[MW22]   MIR DJAWADI, B.; WESTER, L.: Social Proximity and Feedback-Giving Behavior - A study on how social interaction influences feedback-giving behavior on peer-to-peer platforms. In: *Mimeo* (2022)

[RS12]   ROCKENBACH, B.; SADRIEH, A.: Sharing information. In: *Journal of Economic Behavior & Organization* 81 (2012), no. 2, pp. 689–698

[SMH+21]   STRAATEN, D. van; MELNIKOV, V.; HÜLLERMEIER, E.; DJAWADI, B. M.; FAHR, R., et al.: *Accounting for heuristics in reputation systems: An interdisciplinary approach on aggregation processes*. Tech. rep. Paderborn University, Faculty of Business Administration and Economics, 2021

[Sun12]   SUN, M.: How does the variance of product ratings matter? In: *Management Science* 58 (2012), no. 4, pp. 696–707

[TK74]   TVERSKY, A.; KAHNEMAN, D.: Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. In: *science* 185 (1974), no. 4157, pp. 1124–1131

[ZHKN18]   ZIMMERMANN, S.; HERRMANN, P.; KUNDISCH, D.; NAULT, B. R.: Decomposing the variance of consumer ratings and the impact on price and demand. In: *Information Systems Research* 29 (2018), no. 4, pp. 984–1002

Subproject B1:

Dialogue-Based Requirement Compensation and Style-Adjusted Data-To-Text Generation

Frederik S. Bäumer[1], Wei-Fan Chen[2], Michaela Geierhos[3], Joschka Kersting[2], Henning Wachsmuth[4]

1 Applied AI Group, Bielefeld University of Applied Sciences, Bielefeld, Germany

2 Department of Computer Science, Paderborn University, Paderborn, Germany

3 Research Institute CODE, University of the Bundeswehr Munich, Neubiberg, Germany

4 Institute of Artificial Intelligence, Leibniz University Hannover, Hannover, Germany

1 Introduction

OTF computing is exploring ways to provide people with more customized, on-demand software services that meet their needs. Similar to using a search engine, users should be able to express their needs in natural language without any special technical background. For this reason, processing and interpreting natural language requirements is an essential part of the OTF vision. Since formal specifications are not very intuitive, natural language tends to be the only format for service descriptions that most users will consider.

Subproject B1 deals with different types of service requirement specifications that enable a successful search, composition, and analysis of services. In the sense of agile, collaborative software development, the idea is to involve users in an interactive composition process of software services to be created on-the-fly. This setting implies a dialogical situation between users and systems, suggesting the use of a domain-specific chatbot for a specific query on the one hand, and the resolution of ambiguities on the other. For such a composition process to be successful, it should be transparent to users. In particular, it must be clear which initial requirements were taken into account in the creation and which had to be dropped.

Service descriptions can be considered a less formal form of requirement specifications because they describe a service in natural language. Moreover, the accuracy of a description depends on a number of factors, such as the proficiency of the requirement's author. For this reason, service descriptions are sometimes referred to as user-generated informal documents [MLC14]. Formal and semi-formal description languages are one way to avoid the shortcomings of natural language in OTF computing. The software specification language

---

frederik.baeumer@fh-bielefeld.de (Frederik S. Bäumer), cwf@mail.upb.de (Wei-Fan Chen), michaela.geierhos@unibw.de (Michaela Geierhos), jkers@mail.upb.de (Joschka Kersting), h.wachsmuth@ai.uni-hannover.de (Henning Wachsmuth)

is an example of providing comprehensive service specifications and is applicable to both non-functional and functional software requirements [PJR+16]. But even in a simplified form, users cannot apply it as they lack the necessary technical expertise [FdSG14; GSB15]. Therefore, developers must accept free-form, natural language requirement descriptions from users. In doing so, they have to deal with difficulties that are typical for free-form text. These include, for example, a lack of structure and correctness, grammatical and spelling errors, and ambiguity in syntax and semantics. In addition, there will be missing information that is essential for development, but that a user does not have in mind. Thus, callbacks are unavoidable.

Initially, this subproject focused on the development of a parameterized core language for services [Pla13; BBP15] to process requirements automatically, accurately, and efficiently. Due to a lack of acceptance by the target group, user-friendly requirement specifications were proposed as an alternative [Bäu17; BG18]. These specifications are intuitively understandable and mainly used by customers in the OTF market (i.e., end users or domain experts). To improve transparency and human-machine collaboration, dialogue-based requirements compensation was later introduced [KAG22], which provides explanations of services in natural language [CASW21]. Comprehensible explanations are needed because the deficiencies in requirement specifications cannot always be automatically compensated and because users do not know which of their requirements have been fulfilled or not, why this is the case, and what other services need to be included. Figure 10 illustrates the outlined evolution of the process over three funding periods.
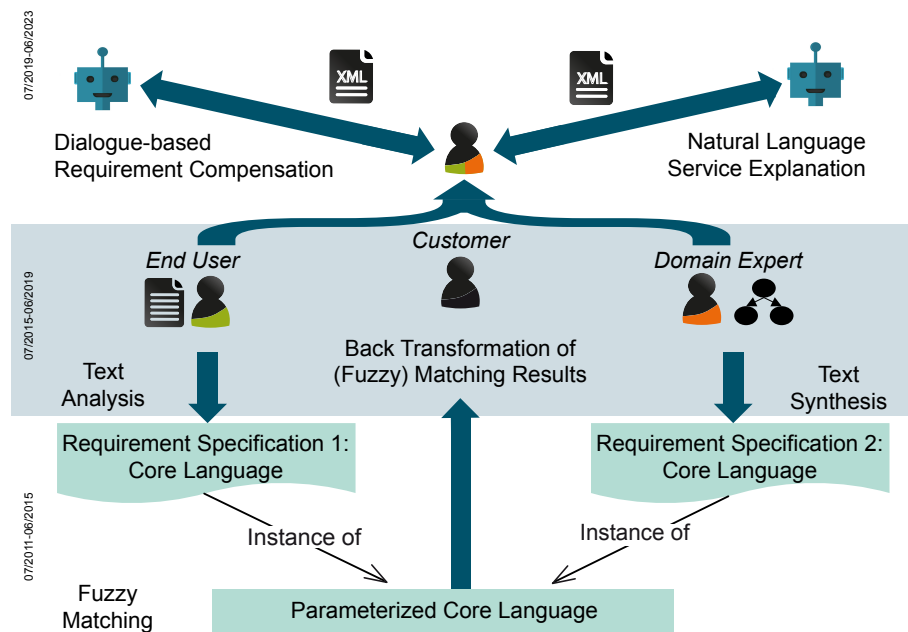
Figure 10: *Overview of the development phases of subproject B1 (bottom to top).*

In the following, we present and discuss four highlights of Subproject B1, the first two of which are related to the *automated optimization* of specifications, the others to the *generation of explanations* of services.

1. **Compensation of linguistic deficiencies:** The automatic selection of techniques that fit the user's description (cf. Sec. 2.1).

2. **Chatbot-enhanced deficiencies resolution:** The interactive resolution of remaining deficiencies in a dialogical manner (cf. Sec. 2.1).

3. **Data-to-text explanation generation:** The computational generation of a text explaining the main features of a created service (cf. Sec. 2.2).

4. **Style adjustment of explanations:** The computational transfer of the text's style to the language of a specific user group (cf. Sec. 2.2).

2   Highlights and Lessons Learned

The research highlights presented in the following contribute to the broad field of natural language processing (NLP). They include novel fundamental computational methods as well as new applications of NLP in real-world technologies for the given scenarios.

2.1   Automated Optimization of Natural Language Specifications

There have been few attempts to address the variety and deficiencies of natural language software specifications as they arise in the software specification process by users without excessive back-checking. On the one hand, we have shown that it is possible to involve the non-specialist requirement creators in the software specification process without limiting their ability to express themselves and to support the software developers by automatically clarifying software specifications. The latter can avoid minor callbacks and thus reach the implementation faster. On the other hand, we addressed the detection of inaccuracies and incompleteness in requirements descriptions that still leave too much room for interpretation for a concrete software implementation. For this purpose, we developed a procedure that compensates ambiguous and partially incomplete statements of the requirement creator by using intelligent (request-driven) knowledge queries.

Compensation of Linguistic Deficiencies in Service Descriptions

Our goal was to develop a parameterized model that automatically chooses the right strategy to compensate for human shortcomings in natural language specification [Bäu17].

Since software descriptions are sensitive to linguistic deficiencies such as ambiguities and elisions, which can delay and thus hinder the specification process, a workaround had to be found. However, there are numerous software programs that can detect and partially correct deficiencies in requirement descriptions. Unfortunately, however, they are often difficult to use and not suitable for end users. In addition, they usually do not cover the full range of flaws and inaccuracies that can occur in natural language [BG18]. Examples include methods that can detect and correct multiple deficiencies in natural language requirements and tools that can be classified as expert solutions. These focus on a single phenomenon of linguistic imprecision, such as lexical ambiguity. However, there are also solutions that can detect ambiguity and incompleteness together or that can find different forms of ambiguity [TB13; Kör14]. Some studies [HB15; SJ15; Bäu17] show that many software solutions focus only on finding deficiencies. Therefore, users are still in charge of

compensation. Furthermore, these methods have not been combined to compensate for service descriptions. End users expect their software requirements to be fully implemented, but are unable to identify and correct linguistic deficiencies in requirements themselves. In addition, there is no guarantee that end users will notice ambiguities or incompleteness, although this can be very frustrating. These problems can be solved by implementing an interactive, computer-aided compensation process [BG18].

With our approach, users do not have to select the necessary techniques to compensate for the deficiencies in their service descriptions; this is done automatically. This minimizes the number of callbacks and thus streamlines the process. As a result, outputs are improved and users are freed from persistent, manual tasks. Finally, our approach improves the state of the art in OTF computing [BG18]. Our research methodology, while following the principles of design science, results in a software tool called CORDULA [Bäu17]. Figure 11 shows the modules used in our text analysis pipeline and their interaction.
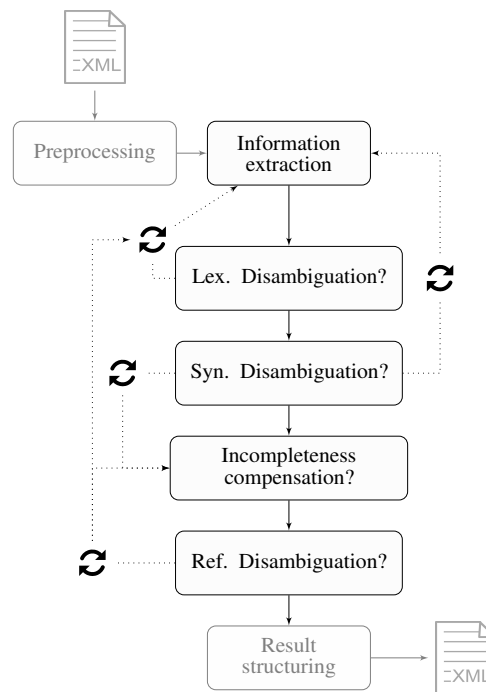

Figure 11: *Indicator-based text analysis pipeline for deficiency compensation [BG18].*

As can be seen in Figure 11, we process the user input step by step, starting at the top. After preprocessing, we extract information. We can deal with syntactic, lexical, and referential ambiguity, as well as incompleteness. The configuration and execution, as well as the monitoring of the processing pipeline, are then performed by an indicator-based compensation strategy [BG18]. The strategy allows to exploit the synergies between the techniques used. However, the most important question here is whether the input is a software description. This pre-step is called on-off-topic classification, and the tool developed for this purpose is called REaCT [DG16]. It finds process words, i.e., semantic information such as the role or the action in a service description. This is done to ensure that the input is indeed a software requirement description. Every other module, including referential disambiguation, is triggered by the corresponding requirement quality indicator [BG18]. Finally, the improved software description is then presented to the user.
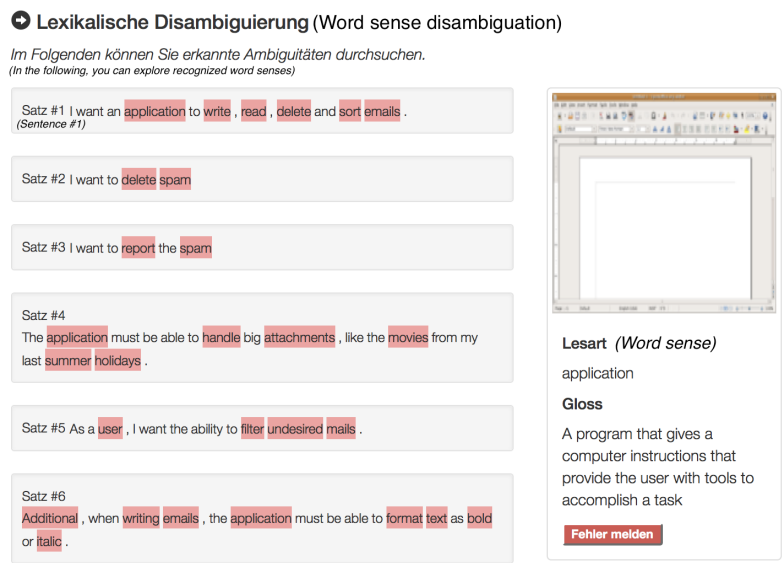
Figure 12: *Lexical disambiguation results as presented by CORDULA [Bäu17].*

Looking more closely at the individual modules of the pipeline, lexical disambiguation, for example, aims to find the correct meaning of a word. After contextualizing a word, the application finds several possible readings. It is triggered when more than one reading is possible. The process involves the REaCT tool to semantically classify the tokens (i.e., to recognize actions, objects, etc.). In this way, readings can be eliminated, and finally, only the disambiguation candidates remain (see Figure 12).
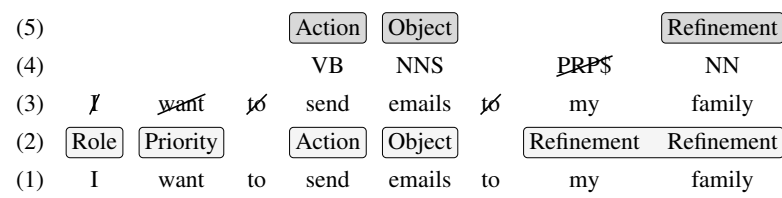
(5)			Action	Object			Refinement	
(4)			VB	NNS	~~PRP$~~		NN	
(3)	~~I~~	~~want~~	~~to~~	send	emails	~~to~~	my	family
(2)	Role	Priority		Action	Object		Refinement	Refinement
(1)	I	want	to	send	emails	to	my	family

Figure 13: *Natural language processing with semantic role labeling [BG18].*

In addition, incompleteness compensation addresses service descriptions that lack information. REaCT's semantic role labeling (see Figure 13) is used to detect incomplete sentence constructs and to identify the missing information. Another ambiguity in language is referential ambiguity, i.e., which pronouns refer to which nouns. This is critical for the system to understand the query as a whole. We used part-of-speech tagging in combination with discourse analysis to solve this problem [BG18].

Based on this concept, we developed a system that uses automatic compensation strategies to assist end users in creating clear and comprehensive natural language requirements. For this purpose, linguistic indicators were developed to identify the need for each compensation method in the descriptions. Based on these indicators, the entire text analysis pipeline we have built is configured ad-hoc and then tailored to the unique details of a service description [BG18]. The technical implementation was done using CORDULA, a tool for the *Compensation of Requirements Descriptions Using Linguistic Analysis* [Bäu17], whose original frontend can be seen in Figure 14.

Ihre Eingabe *(Your input)*

Hello Marcel! I want an application to write, read, delete and sort emails. I want to delete spam and I want to report the spam.

Ergebnis *(Result)*

Nr.*(No.)*SID		Anforderung *(Requirement)*
1	S1	❯ As a user, I want to write emails ❯ As a user, I want to read emails ❯ As a user, I want to delete emails ❯ As a user, I want to sort emails ☑ I want an application to write, read, delete and sort emails.
2	S2 ✂	❯ As a user, I want to delete spam ☑ I want to delete spam 📄 I want to delete spam and I want to report the spam.
3	S3 ✂	❯ As a user, I want to report spam ☑ I want to report the spam 📄 I want to delete spam and I want to report the spam.

Figure 14: *Frontend for tracing the processing steps of the text analysis pipeline [Bäu17].*

CORDULA allows users to express their software requirements in natural language. It checks the descriptions for weaknesses such as incompleteness and ambiguity, but also for flaws, based on the pipeline shown in Figure 11. The tool compensates for these deficiencies in an understandable way via a web interface. Figure 14 shows that the output is in a simple language so that users can easily understand the information provided and check its consistency. CORDULA is based on two views: the end-user (and administrator) view and the system view. The frontend shown here is the (simplified) user view. For special information, e.g., disambiguation, additional interfaces are available to help debug the output. After compensation, the prototype transforms functional requirements into structured output for further use in the OTF computing scenario. Thus, others can build on our output, for example, by configuring software services ad-hoc and providing the software requested by the end user [FBG18].

In this highlight, we showed that inaccuracies in software descriptions can be automatically detected and compensated without user interaction. Our processing pipeline is optimized by linguistic indicators, minimizing runtime and user interaction. We also demonstrated that rule-based indicators accurately handle most linguistic deficiencies [BG18].

Chatbot-Enhanced Requirement Deficiencies Resolution

In order to eliminate requirement deficiencies that cannot be compensated by existing methods, dialogues are required that help to specify and complete derived requirement specifications. Existing extraction and compensation methods generate preliminary service templates from which software services can be selected and composed. The goal of dialogue control is to iteratively revise deficient templates in a guided manner in order to provide as much information as possible for service composition. To achieve this, we have divided the dialogue control into a requirement interpretation and a chat interpretation. While the latter is responsible for the dialogue control and the interpretation of all user input, the requirement interpretation is responsible for the classification, interpretation, and
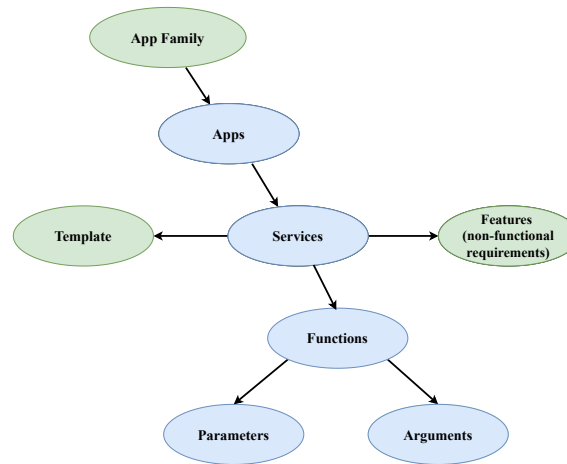
validation of detected requirements. This separation makes it possible to integrate existing chatbot techniques, while the extraction and compensation components developed in previous work can be used for processing software requirements. Especially for end users with little prior technical knowledge, it is necessary to perform the iterative clarification processes not only in dialogue but also with the help of explanations and examples. For example, when compensating for incompleteness, this may mean not only pointing out the missing information and asking for it to be filled in, but also providing a similar example that fits the context [BKG19].

In [FBG18], CORDULA has been further developed to take requirements deficiency resolution to a new level. The goal here is to maximize the automation of the process. It has been shown that domain-specific resources are needed to produce high-quality results. In addition, we have found that it is necessary to involve users in the process instead of presenting them with the results and having them go through the process again if they are not satisfied [BKG19]. Consequently, we developed a new approach for CORDULA that includes a chatbot and a knowledge base. Both support the given process by clarifying the possibilities and limitations of the software configuration process to the user, and the chatbot asks if the information is incomplete or ambiguous [Ahm22; KAG22].

Figure 15: *Bidirectional information flow in the enhanced CORDULA version.*

Figure 15 shows our evolved solution, which, unlike its predecessor [Bäu17] (cf. Figure 11), is bidirectional. As you can see on the left side, we have a dialogue flow where the user interacts with an automated chatbot system. In the middle are the relevant elements of the processing pipeline. On the right is the knowledge base. The natural language processing pipeline shown in Figure 11 has been further developed and integrated into our solution, which includes on-off topic classification and incompleteness detection, but also new features.

Based on the use of domain knowledge, we also created a deep learning model that recognizes app families, which are stored in the knowledge base [Ahm22]. Figure 16 shows a snippet of the structure of this knowledge base. There is a hierarchical order describing app families, corresponding apps, services, functions, and templates. By using the knowledge base, the chatbot knows exactly what information it needs from the user to fill out a service template. In addition, the chatbot uses the app family to decide which path options are possible, which templates, and which service descriptions are possible.

Figure 16: *Graph representation of a substructure of the knowledge base [Ahm22].*

For the dialogue management of the chatbot, we used Rasa[5] after testing different solutions such as DialoGPT [ZSG+20] or ChatterBot[6] (see Table 1).

Models	Effectiveness	Efficiency	Satisfaction	User Approval
Rasa & knowledge base	0.84	0.78	0.73	0.73
ChatterBot	0.26	0.21	0.31	0.36
DialoGPT	0.47	0.79	0.52	0.63

Table 1: *Performance ratings for tested chatbot systems [Ahm22].*

Table 1 presents the effectiveness results in terms of the conversational intelligence and performance shown by the chatbot with respect to the user's goal. Efficiency refers to whether the chatbot seemed human-like. Satisfaction captures whether the user is satisfied with the chatbot's responses and behavior (e.g., politeness). User approval deals with the overall satisfaction of the user. DialoGPT cannot be directly combined with a knowledge base; the knowledge should be incorporated into the parameters of the chatbot. The chatbot achieves partially convincing results. However, ChatterBot is not convincing at all. We conclude that Rasa is the best solution based on an evaluation of user interactions. The intent classification works well, while the dialogue management allows us to add features from CORDULA and integrate the chatbot into our knowledge base. When users tested all three implementations, Rasa was by far the best in terms of overall performance, but also in terms of fine-grained evaluation results [Ahm22; KAG22].

Figure 17 shows an example of a user-chatbot interaction. Here, the resolution of the requirement description is done in a bidirectional manner. The chatbot greets the user and asks for the user's requirement descriptions. Each time the response is successfully processed by the system, the chatbot provides this as feedback ("Your requirements were successfully processed."). Some of the responses follow a rule-based approach to ensure that the requested information is always provided to each user. After the chatbot receives and processes the initial user input, it asks for more information. In the backend, the system queries the knowledge base and performs matching tasks using deep learning models to

---

[5]https://rasa.com/, last accessed 2023-02-06.
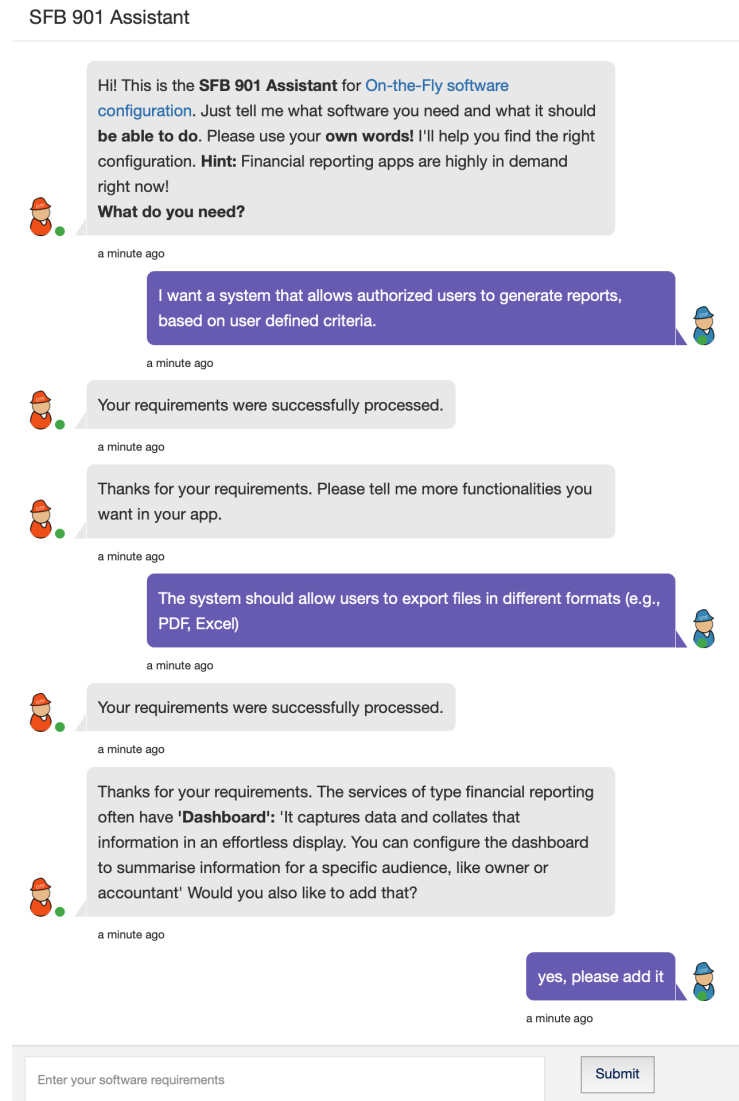[6]https://chatterbot.readthedocs.io/en/stable/, last accessed 2023-02-06.

Figure 17: *Example of a conversation with the enhanced CORDULA version [Ahm22].*

look for relevant service configurations or features that already exist in the OTF computing market. The user is free to decide what features to include or not include in the app. When there is enough information about the app that the chatbot has collected together with a user, the chatbot gives the user a summary of the requirements that have been met and those that have not. This way, the user knows what to expect from the configured app provided by the OTF system [Ahm22; KAG22].

In addition, the chatbot deals with parameters that are initially unknown to the user. This means that software configurations sometimes have requirement templates, for example, because some services cannot be combined with others, or because some apps require certain services. End users cannot know this, but they also do not have to: The system tries to fill in the templates in the background during the conversation and controls the chat accordingly, e.g. by asking appropriate questions and suggesting certain services. The aforementioned summary, which is communicated before the app is deployed, also includes services that have been added based on a template.

In summary, our research has solved the following three main problems: (1) Deficiencies in requirement descriptions cannot always be automatically fixed, which we address through our bidirectional chatbot solution with a knowledge base. (2) Configuring parameters that are initially unknown to users leads to errors in the service templates. We integrated this into the conversation, and we provided explanatory summaries. (3) Previously, users did not know which of their requirements were met (and which were not) in a created service, they had to start from scratch. This is also solved by our summary and explanation at the end of the chat. The user sees a list of all requirements, whether they were met, and where they came from (from the user, from a template, or from an app/service suggestion accepted by the user).

2.2   Natural Language Explanation Generation

One of our key goals was to make the service configuration process more comprehensible for end users. In this way, users should not have to find out by trial and error whether their service configuration expectations were met. Instead, they should get an early insight into the feasibility of their requirements. For this purpose, we investigated the extent to which typical user formulations based on the requirement descriptions are suitable for training a text generation approach and which techniques can be used for text generation in a dialogical question-answer setting.

To improve user understanding and service transparency, we provide explanations in natural language that describe the created service while being adjusted to the user's language. Human-like explanations are still understudied in natural language processing (NLP) research [WA22], particularly their generation. To obtain correct but understandable explanations, we combine the results of two complementary efforts: On the one hand, we developed a data-to-text generator for natural language explanations and evaluated its performance on a data set of explanations. On the other hand, we investigated how to adapt the linguistic style of the generated explanations to the user's proficiency. For the latter, we developed novel text style transfer methods and evaluated them on a corresponding data set. Both are described in detail below.

Data-To-Text Stylized Generation of Service Explanations

In the following, we summarize the contributions that we have made to data-to-text explanation generation with a touch on style adjustment [Bül21; ACGW21]. Specifically, we have attempted to generate natural language sentences from the output of the services. Here, the output is given as a set of tuples of the $< name, value >$. For example, $< F_1, 0.89 >$ declares the F_1-score of a computational model used in the service to be 0.89. In NLP, the underlying task is known as a data-to-text generation problem.

Since we did not have a sufficient number of service outputs with the target sentence data, we decided to study the given problem on data from another domain that shares similar properties with service explanations at an abstract level. In particular, we make use of transcriptions of politicians' speeches. These speeches naturally contain explanations of policies, actions, and the like. From OTF computing perspectives, such explanations provide the data we are interested in by using a distant-supervision manner.

A specific aspect of interest within the given task is the consideration of different linguistic styles. In our study, we collected speeches from various politicians and conducted experiments on them. For a controlled setting, we focused on the styles of two former U.S. presidents, Barack Obama and Donald Trump. Both have clearly recognizable speaking styles. Learning to generate texts that match the styles of the two presidents is challenging. Specifically, it involves at least the following types of style adjustment:

1. **Formality:** The complexity of the words used by the two presidents also differs significantly. In line with expectations, we observed that Obama uses more professional and educated words, while Trump prefers simple language. Analogous to our service explanation task, formality also serves as the generated explanations' formality.

2. **Political bias** and **subjectivity:** The two presidents are from various parties. As a result, they support or oppose different policies and have opposing views on several issues. For example, Obama is seen as being open to immigrants, unlike Trump. Our analysis also revealed that the subjectivity of the two presidents' language plays a role in distinguishing their styles. For example, Trump uses many emotional words in his speeches, while Obama avoids them. These two style features are specific to political speeches. In our service explanation task, it helps to specify word usage preferences in the explanation, such as preferring F-score instead of F-measure.

In our research, we investigated how different models learn and adapt these different aspects. For this purpose, we built a data set with a total of 962 transcribed public speeches, press conferences, and interviews of the two presidents, 434 of Obama and 528 of Trump. Since the two presidents' speeches have limited overlap in time, the data set is naturally non-parallel, meaning that we had to learn style adjustment without having training pairs from which to infer style correspondences directly.

Given the data set, we focused on *sentence-level* style adaptation as a first substantial step towards full document generation. First, we analyzed the training set to understand the style of each politician. In line with our previous work [CWAS18], we calculated the most discriminative words of each style to analyze each former president's word usage preference, and we determined the proportion of emotional words in the speeches. The results highlighted that Trump frequently used emotional words (e.g., disgrace or lied), while Obama did not (e.g., resolved or urgency). This informed us about what to look for when developing and evaluating style transfer approaches. For example, after a style transfer, the text should have a similar distribution of word usage as in the desired style. We also found that not all sentences in the speeches had significant style indicators. Therefore, we used only those sentences that contained at least one discriminative word (words used twice more often in one style) found by the analysis.

On this basis, we evaluated two approaches to the explanation generation task:

1. **NER + Data-to-text:** This is the approach we developed. It first extracts named entities in a sentence before applying one of two (one for Obama and one for Trump) fine-tuned neural language models (BART) to generate a sentence from the given named entities. The NER step is designed to capture the main content of the sentence, while the data-to-text part is designed to generate a sentence about the named entities using a specific speaker style.

2. **Cross-aligned autoencoder:** For comparison, we used a cross-aligned autoencoder for this task [SLBJ17] as a baseline. This model learns to optimize two neural

	Style Adaptation		Explainability	
	Automatic	Manual	Automatic	Manual
Cross-aligned autoencoder	68%	2.75	44%	2.24
Data-to-text generation (our approach)	86%	3.55	13%	1.62

Table 2: *Evaluation of data-to-text generation in terms of style adaptation and explainability: automatically computed percentage of success and manually assigned scores from 1 (worst) to 5 (best) for the cross-aligned autoencoder and for our approach.*

autoencoders to separate content and style in a latent space. An important feature of the model is that it can preserve the text structure and change the style of the text. The cross-aligned autoencoder shows how we can generate the explanations and adapt the style using a model.

The approach we developed for the explanation generation task first extracts named entities from the political speeches as the targets of the explanation. These named entities include the names of policies, organizations, and politicians. After extracting the named entities, the second step is to train a data-to-text model to generate texts from the extracted named entities, as shown in Figure 18.
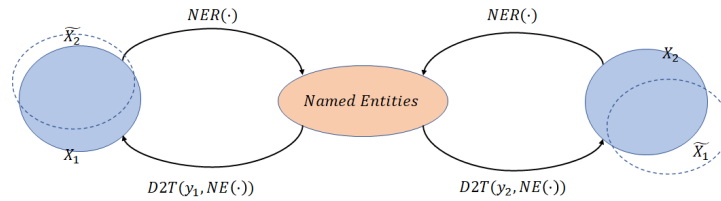

Figure 18: *The data-to-text explanation generator by [Bül21]. The approach is a combination of named entity recognition (NER) and a data-to-text generator (D2T).*

The idea of our approach is to express the content of a sentence in political speaker transfer only on the basis of the named entities. All other words are considered as style words by the speaker. The three key elements of our approach are as follows.

First, the *encoding* is the reduction of a sentence to its textual keywords. This is done using a widely used information extraction algorithm, namely the NER system of the popular Python library spaCy. Second, instead of a latent *representation*, the input is textual. It consists of a number of instances, each composed of the named entities of a proposition and their respective types. This has the advantage of being explicable, which is often not the case with neural network models. And third, *decoding* is the process of transforming the named entities into an explanation using natural language, relying on the outlined data-to-text transformer model. The model, trained on a corpus of either political speaker, should implicitly learn to generate an explanation that expresses the speaker's style and thus resembles the language usage.

We automatically and manually evaluated our approach and the baseline. Both evaluations were aimed at the effectiveness of style adaptation and explainability. The results are shown in Table 2. The automatic results are given as a percentage of success, while the

manual results are scores ranging from 1 to 5, with 5 being the best. Table 2 suggests that both approaches have pros and cons, namely that the cross-aligned autoencoder has better explainability, while our approach succeeds in style adaptation.

The following sentences are examples generated by our approach to describe how the two politicians talk about Europe and China:

> Trump style: *They'll compare us to Europe and we did very well.*
> Obama style: *That's the spirit that binds us to Europe.*
>
> Trump style: *If Biden is elected, China will own America.*
> Obama style: *China is firmly committed to the path of peaceful development.*

In these two examples, the Trump-style sentences convey negative opinions about Europe and China. The Obama-style sentences describe the two entities from a more neutral point of view. This phenomenon suggests that our approach learns the language use of the two politicians in terms of the usual way they describe other countries, where Trump typically emphasizes the competition while Obama focuses on the cooperation between the countries.

Our qualitative results suggest that the cross-aligned autoencoder often successfully adapts the style by using stylized words, but fails to preserve the explanation sufficiently. In contrast, our proposed approach often successfully reformulates the content in different words. We also observe that after applying this approach, a pre-trained style classifier predicts that the generated text has the desired style.

In conclusion, this section has summarized our contributions based on our experiments on political speeches as a distance-supervision data to study service explanations: (1) We have studied approximately how to generate natural language explanations of services by considering the problem as a data-to-text task. (2) We demonstrated that our approach can generate explanations for the desired language style.

Text Professionalization as an Example of Style-Adjusted Generation

After generating the explanations, the next goal is to adjust the language style of a given explanation, which is called the *text style transfer* task in NLP. Unlike the previous task, where the input was a tuple of $< name, value >$, here the input is a natural language sentence ready to be style adjusted.

In text-style transfer, the goal is to rewrite a text in a defined style while keeping the content of the text similar. Text-style transfer tasks have become increasingly popular in the era of deep learning. Depending on the goal of the task, the term "style" can refer to different attributes of a text, such as media bias [CWAS18] or the speaking style of politicians as mentioned above.

In the following, we summarize our main findings from two text-style transfer studies on a specific task that exemplifies the adjustment of a text to a specific proficiency level [Mis21; Pal22; CASW20]:

> **Text professionalization:** Given a text, rewrite it so that its style becomes more formal while preserving as much of its original content as possible.

The following two sentences show an example of how a text in "normal style" can be rewritten in a more professional way while still conveying the same information:

> Normal: *Oktoberfest happens every year in Munich and many people take part in it, where they drink beer at the Theresienwiese square.*
>
> Professional: *The Oktoberfest occurs annually in the German city of Munich and is celebrated by a large crowd, where participants gather in beer-drinking festivities in the Theresienwiese square.*

The professional sentence in the example above uses synonyms for "happens" and "every year" that reflect a more sophisticated style ("occurs" and "annually"). It also provides more context by adding the country in which the city is located ("the German city of Munich"). Finally, the professional sentence uses other phrases, such as "is celebrated by a large crowd" and "participants gather in beer-drinking festivities," which enrich the sentence with elaborate details compared to the standard phrases "many people take part in it" and "they drink beer."

For our research, we used SSCORPUS [KK16]. This corpus has 493,000 aligned sentences extracted by pairing simple English Wikipedia[7] with standard English Wikipedia[8]. It contains sentences from Wikipedia articles on various topics, which fits our research goal of text professionalization. Each instance in the corpus is composed of a sentence from standard Wikipedia, the corresponding sentence from simple Wikipedia, and a similarity score between them. The sentence from standard Wikipedia uses formal and professional language that fits the characteristics of how professional language should be. It is followed by a simpler version of the same sentence extracted from the simple Wikipedia, which uses simple and regular words that are easier to understand. Finally, the similarity score, ranging from 0 to 1, describes the similarity in meaning of the two sentences: A score of 1 means that the two sentences are identical, while a score of 0 means no similarity at all.
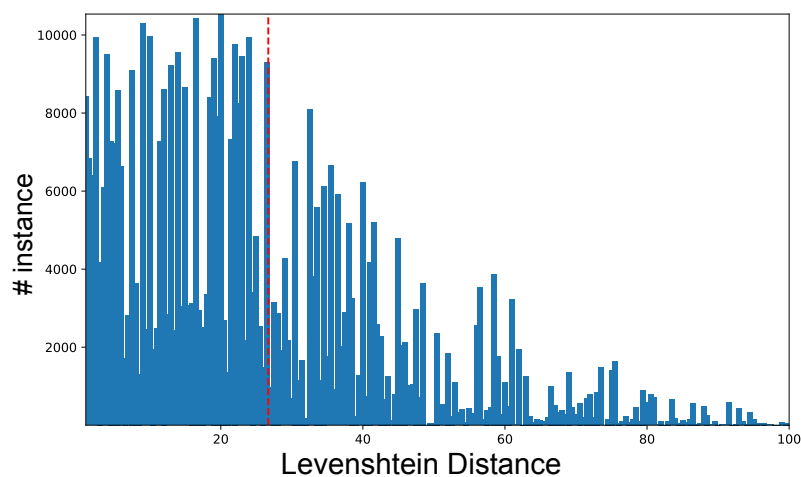

Figure 19: *The Levenshtein distance distribution in SSCORPUS. The red line marks the threshold we used for filtering.*

The first step was to filter the given data set. In particular, the original use of SSCORPUS

---

[7]`https://simple.wikipedia.org`, last accessed 2023-02-06.
[8]`https://en.wikipedia.org`, last accessed 2023-02-06.

	Automatic Evaluation		Manual Evaluation		
	BLEU	ROUGE	Factuality	Fluency	Professionality
Simple	-	-	4.07	3.94	3.95
Professional	-	-	4.16	4.03	4.01
Generated	0.59	0.64	4.21	4.07	4.07

Table 3: *Evaluation of style-adjusted generation: automatic scores (BLEU, ROUGE) and manually assigned scores from 1 (worst) to 5 (best) in terms of factuality, fluency and professionality for the simple sentence, the professional sentence, and the generated sentence.*

is to transfer the text from its simple version to its professional version. However, we could not simply switch the inputs and outputs of SSCORPUS, because the professional versions of the text not only use more professional words but also add more details. In other words, transferring to simple texts results in removing these details, while transferring to professional texts means adding these details. In practice, it is easier to remove information than to add it. Sometimes it is almost impossible to add extra information out of nothing. As a measure, we used the Levenshtein distance between the simple and professional versions and discarded those pairs that added too much new information (having a too-high Levenshtein distance). Figure 19 shows the distribution of the Levenshtein distance in SSCORPUS and the threshold (25) for selecting the better instances. In order to obtain a professionalized text, we used the context of the text as an additional input in our approach. Specifically, we used entity information to model the context, where the entity information was extracted using the NER tool spaCy. Figure 20 shows the architecture of the model within our approach.
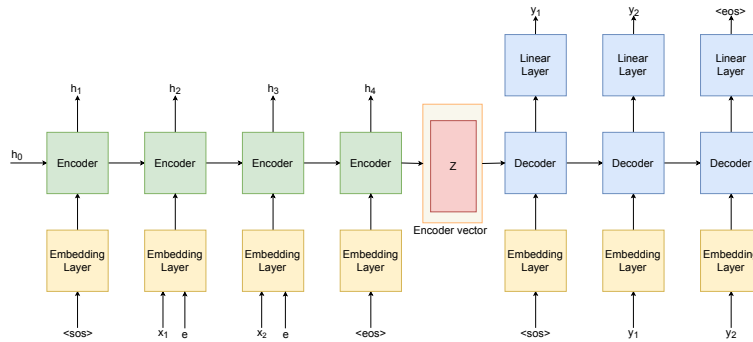

Figure 20: *The architecture of our model adapted from Mishra [Mis21]: x_n and y_n denote the n-th input and output tokens, h_t is the hidden state in the step t, <sos> and <eos> are the start and end of sentence tokens, z is the representation of the input, and e is the entity information as an additional input to the model.*

The final step was to select a base model for fine-tuning on our data set. We chose BART from Facebook AI, which was pre-trained on the CNN/DailyMail data set. At the time of developing our approach, BART showed state-of-the-art performance on many NLP tasks, such as machine translation, question answering, and text simplification.

We re-evaluated our approach using both automatic and manual measures. In the automatic evaluation, we considered BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation). For manual evaluation, we used Amazon Mechanical Turk to evaluate the text in terms of fluency, factuality, and professionality. The evaluation results are shown in Table 3. The BLEU and ROUGE scores indicate that the generated sentences are close to professional sentences. In terms of manual evaluation, the judges gave very high scores to the generated sentences in all three aspects, indicating that our approach successfully transforms simple sentences into professional sentences.

The following is an example of a professional sentence generated by our approach:

> Simple: *During pregnancy, the endometrium develops a lot of glands and blood vessels.*
>
> Professional: *During pregnancy, the glands and blood vessels in the endometrium further increase in size and number.*
>
> Generated: *During pregnancy, the endometrium develops a large number of glands and blood vessels, known as endometrial granules.*

Here, the professional sentence changes the part "develops a lot of" into a more precise description that the endometrium "increase in size and number." The generated sentence also professionalizes the simple text in a similar manner by changing it to "develops a large number." Furthermore, the generated sentence adds a new concept "endometrial granules." The model appears to learn such word usage from the pre-trained weights of the BART model. In addition, a judge from Amazon Mechanical Turk commented that *"all sentences are good, but the generated one is the clearest and most descriptive."*

In summary, this research investigates one of the core topics of Subproject B1: adjusting a text to the user's language. Depending on the professionality of the users, our approach can transfer the style of the text to a more professional version. In principle, it could also be easily adapted to transfer the texts to a less professional version. From the OTF computing perspective, we study how to generate explanations for different user groups. For example, machine learning beginners would be simple explanations with few technical terms. On the other hand, experts would expect explanations with algorithm details.

3   Impact and Outlook

Our work is embedded in the context of OTF computing where our goal was to allow end users to participate in the specification process by supporting natural language requirements without limiting expressiveness. The richness and ambiguity of natural language, as well as its imprecision, may cause deficient service descriptions and hence posed a challenge for processing, which had to be mastered. By incorporating natural language generation and style transfer techniques, we enabled communication with users in a language they can understand.

3.1   Optimization of Service Specifications

Until now, most of the existing work has focused on the use and development of semi-formal or formal specification languages, which means that there have been few approaches

in the area of requirements extraction. In particular, there has been a lack of linguistic resources. To be able to process natural language requirements, we developed our own knowledge-based resources (e.g., requirements corpora and compensation graphs [GB16]) and an approach (REaCT, Requirements Extraction and Classification Tool) that detects on-topic statements in service descriptions. Based on this, the insufficiently specified end-user requirements could be analyzed (i.e., identified, extracted, and formalized) to compensate for ambiguity, vagueness, and incompleteness.

In order to efficiently and reliably compensate for deficient service descriptions, strategies have been developed to select appropriate algorithms [GB17]. The goal was to obtain concrete natural language service specifications depending on certain linguistic inaccuracies (e.g., ambiguity and incompleteness) in order to best specify the user's original requirements. For this purpose, a parameterized model has been developed [BG18] that automatically chooses the most appropriate strategy to compensate for linguistic deficiencies in service descriptions. On the one hand, strategies were developed for demand-oriented and performant control of appropriate compensation procedures. On the other hand, it was shown that the strategy configuration itself can be performed in a data-driven manner in real-time, depending on the situation, and can lead to better results than predefined rule sets. Also, the proven learning effects of the prototype CORDULA (Compensation of Requirements Descriptions Using Linguistic Analysis) [FBG18] through caching during (domain-specific) lexical disambiguation could be achieved in practice. Furthermore, in line with our earlier research [WSE11], we questioned the predefined order of text analysis steps that has been established in natural language processing for years and achieved good results by deviating from the classical NLP pipeline. In this way, the respective input text itself reveals which inaccuracies are present, and the necessary compensation steps as well as their order are determined in a data-driven manner, taking into account the interactions between ambiguous and incomplete linguistic expressions.

Based on this work, we moved to bidirectional requirement compensation, since unidirectional and analytic approaches do not produce fully actionable requirements. Therefore, we used the previously constructed processing pipeline and enriched it with a chat functionality that can access an underlying knowledge graph created specifically for the domain. The chatbot ensures that user requests are processed end-to-end. With access to the knowledge graph, the chatbot suggests application features to the user, fills out templates in the background, and asks questions that help the chatbot determine the type of application being requested. After evaluating this approach with real users, we will be able to generate software requests based on natural language [KAG22].

3.2 Generation of Service Explanations

Learning to explain sophisticated concepts is never an easy task, even for humans. In our research, we faced two major challenges: First, we needed data sets to analyze human explanations and to train explanation generation methods. Second, we wanted to adapt the style of the explanations to the language of the users.

We decided to use data sets from other task domains because no service-related explanation data were available. In the first highlight we presented in Section 2.2, we outlined how we adapted the transcriptions of politicians' speeches into a data set for explanations. In this

way, we were able to collect the explanations remotely, rather than having to ask experts to write the explanations for our study. In the second highlight, we wanted to provide users with different professional levels of explanations. Again, no perfectly matched data were available. To solve this problem, we used the existing SSCORPUS from another task and further filtered it to meet our needs for text professionalization. In both highlights, we described how to solve the problem of data sparseness and also demonstrated that we can use the data sets created in the two highlights to properly study the desired research topics.

The second challenge was to adjust the style of the explanations. In the two highlights, we have shown that such text style transfer can be done in two different ways: On the one hand, we trained multiple data-to-text explanation generators, one for each style; in the example of politicians' explanations, we had two generators, one for each politician. On the other hand, we trained a neural style transfer model and applied it to the explanation results. In the task studied, the professionality of the texts was successfully changed after the texts were generated. Both approaches have their advantages and disadvantages. In the first solution, we trained the model to perform explanation generation and style adjustment in one approach, i.e., as an end-to-end model. In the second solution, we treated the two tasks separately, suggesting a cascading architecture. In general, an end-to-end approach may achieve better performance, while a cascade approach may suffer from error propagation. However, the cascade approach may be easier to interpret, and one can optimize each component individually.

In summary, in the two highlights we have discussed how we approached the task of style-adjusted explanation generation. With our approaches, we contribute to Subproject B1 and fulfill one of its main goals: user-adjusted explanation generation.

3.3   Outlook on Explainable Results

Since there are different ways of elaborating software requirements, the question of explainability arises when many improvements have been made. That is, the reasons for correcting certain parts of the requirements and the reasons for the improvements must be clear to the users. Thus, end users should be informed so that they understand the composition of the service and are satisfied with its features. To date, corrections are sometimes highlighted or compared to the input, but there is no explanatory information about the changes. In the OTF context, there are several challenges: Natural language input from end users primarily shapes the software composition, although the nature of the composition (the actual software service selection step) also has an impact. The result is a transparent composition process that includes natural language inaccuracy detection and compensation methods to improve comprehension. To what extent ChatGPT [9] can be useful and adapted for this domain-specific task remains to be investigated.

---

[9] https://openai.com/blog/chatgpt/, last accessed 2023-02-05.

Bibliography

[ACGW21] ALSHOMARY, M.; CHEN, W.-F.; GURCKE, T.; WACHSMUTH, H.: Belief-based Generation of Argumentative Claims. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 224–233

[Ahm22] AHMED, M.: Knowledge Base Enhanced & User-centric Dialogue Design for OTF Computing. MA thesis. Paderborn University, Germany, 2022

[Bäu17] BÄUMER, F. S.: Indikatorbasierte Erkennung und Kompensation von ungenauen und unvollständig beschriebenen Softwareanforderungen. PhD thesis. Paderborn University, Germany, 2017

[BBP15] BÖRDING, P.; BRUNS, M.; PLATENIUS, M. C.: Comprehensive Service Matching with MatchBox. In: *Proceedings of 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 974–977

[BG18] BÄUMER, F. S.; GEIERHOS, M.: Flexible Ambiguity Resolution and Incompleteness Detection in Requirements Descriptions via an Indicator-based Configuration of Text Analysis Pipelines. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*. 2018, pp. 5746–5755

[BKG19] BÄUMER, F. S.; KERSTING, J.; GEIERHOS, M.: Natural Language Processing in OTF Computing: Challenges and the Need for Interactive Approaches. In: *Computers* 8 (2019), no. 1, pp. 1–14

[Bül21] BÜLLING, J.: Political Speaker Transfer–Learning to Generate Text in the Styles of Barack Obama and Donald Trump. MA thesis. Paderborn University, Germany, 2021

[CASW20] CHEN, W.-F.; AL KHATIB, K.; STEIN, B.; WACHSMUTH, H.: Detecting Media Bias in News Articles using Gaussian Bias Distributions. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 4290–4300

[CASW21] CHEN, W.-F.; AL KHATIB, K.; STEIN, B.; WACHSMUTH, H.: Controlled Neural Sentence-Level Reframing of News Articles. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. ACL, Nov. 2021, pp. 2683–2693

[CWAS18] CHEN, W.-F.; WACHSMUTH, H.; AL-KHATIB, K.; STEIN, B.: Learning to Flip the Bias of News Headlines. In: *Proceedings of the 11th International Conference on Natural Language Generation*. ACL, Nov. 2018, pp. 79–88

[DG16] DOLLMANN, M.; GEIERHOS, M.: On- and Off-Topic Classification and Semantic Annotation of User-Generated Software Requirements. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. ACL, Nov. 2016, pp. 1807–1816.

[FBG18] FRIESEN, E.; BÄUMER, F. S.; GEIERHOS, M.: CORDULA: Software Requirements Extraction Utilizing Chatbot as Communication Interface. In: *Joint Proceedings of REFSQ-2018 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 23rd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018)*. Vol. 2075. CEUR-WS.org, 2018

[FdSG14] FERRARI, A.; DELL'ORLETTA, F.; SPAGNOLO, G. O.; GNESI, S.: Measuring and Improving the Completeness of Natural Language Requirements. In: *Requirements Engineering: Foundation for Software Quality*. Ed. by SALINESI, C.; WEERD, I. van de. Springer, 2014, pp. 23–38

[GB16] GEIERHOS, M.; BÄUMER, F. S.: How to Complete Customer Requirements: Using Concept Expansion for Requirement Refinement. In: *Proceedings of the 21st Int. Conf. on Applications of NL to Information Systems (NLDB)*. Ed. by MÉTAIS, E.; MEZIANE, F.; SARAEE, M.; SUGUMARAN, V.; VADERA, S. E. Vol. 9612. LNCS. Springer, 2016, pp. 37–47

[GB17] GEIERHOS, M.; BÄUMER, F. S.: Guesswork? Resolving Vagueness in User-Generated Software Requirements. In: *Partiality and Underspecification in Information, Languages, and Knowledge*. Ed. by CHRISTIANSEN, H.; JIMÉNEZ-LÓPEZ, M. D.; LOUKANOVA, R.; MOSS, L. S. Partiality and Underspecification in Information, Languages, and Knowledge. Cambridge Scholars Publishing, 2017. Chap. 3, pp. 65–108

[GSB15]   GEIERHOS, M.; SCHULZE, S.; BÄUMER, F. S.: What did you mean? Facing the Challenges
           of User-generated Software Requirements. In: *Proceedings of the 7th ICAART*. Ed. by
           LOISEAU, S.; FILIPE, J.; DUVAL, B.; HERIK, J. van den. Special Session on PUaNLP 2015.
           SCITEPRESS, 2015, pp. 277–283

[HB15]   HUSAIN, S.; BEG, R.: Advances in Ambiguity less NL SRS: A review. In: *Proceedings of
           ICETECH 2015*. Mar. 2015, pp. 221–225

[KAG22]   KERSTING, J.; AHMED, M.; GEIERHOS, M.: Chatbot-Enhanced Requirements Resolution for
           Automated Service Compositions. In: *HCI International 2022 Posters*. Ed. by STEPHANIDIS,
           C.; ANTONA, M.; NTOA, S. Vol. 1580. CCIS. Springer, 2022, pp. 419–426

[KK16]   KAJIWARA, T.; KOMACHI, M.: Building a monolingual parallel corpus for text simplification
           using sentence similarity based on alignment between word embeddings. In: *Proceedings
           of COLING 2016*. 2016, pp. 1147–1158

[Kör14]   KÖRNER, S. J.: RECAA - Werkzeugunterstützung in der Anforderungserhebung. PhD thesis.
           KIT, Feb. 2014

[Mis21]   MISHRA, A.: Computational Text Professionalization using Neural Sequence-to-Sequence
           Models. MA thesis. Paderborn University, Germany, 2021

[MLC14]   MOENS, M.-F.; LI, J.; CHUA, T.-S., eds.: *Mining User Generated Content*. CRC Press, 2014

[Pal22]   PALUSHI, J.: Domain-aware Text Professionalization using Sequence-to-Sequence Neural
           Networks. BA thesis. Paderborn University, Germany, 2022

[PJR⁺16]   PLATENIUS, M. C.; JOSIFOVSKA, K.; ROOIJEN, L. van; ARIFULINA, S.; BECKER, M.; ENGELS, G.;
           SCHÄFER, W.: *An Overview of Service Specification Language and Matching in On-The-Fly
           Computing (v0.3)*. Technical Report. HNI, Paderborn University, Germany, 2016

[Pla13]   PLATENIUS, M. C.: Fuzzy Service Matching in On-the-fly Computing. In: *Proceedings of
           the 2013 9th Joint Meeting on FSE*. ESEC/FSE'13. ACM, 2013, pp. 715–718

[SJ15]   SHAH, U. S.; JINWALA, D. C.: Resolving Ambiguities in Natural Language Software Re-
           quirements: A Comprehensive Survey. In: *SIGSOFT Software Engineering Notes* 40 (Sept.
           2015), no. 5, pp. 1–7

[SLBJ17]   SHEN, T.; LEI, T.; BARZILAY, R.; JAAKKOLA, T.: Style transfer from non-parallel text by cross-
           alignment. In: *Advances in Neural Information Processing Systems*. Curran Associates,
           Inc., 2017, pp. 6833–6844

[TB13]   TJONG, S. F.; BERRY, D. M.: The Design of SREE – A Prototype Potential Ambiguity
           Finder for Requirements Specifications and Lessons Learned. English. In: *Requirements
           Engineering: Foundation for Software Quality*. Ed. by DOERR, J.; OPDAHL, A. L. Vol. 7830.
           LNCS. Springer, 2013, pp. 80–95

[WA22]   WACHSMUTH, H.; ALSHOMARY, M.: "Mama Always Had a Way of Explaining Things
           So I Could Understand": A Dialogue Corpus for Learning to Construct Explanations.
           In: *Proceedings of the 29th International Conference on Computational Linguistics*.
           International Committee on Computational Linguistics, Oct. 2022, pp. 344–354

[WSE11]   WACHSMUTH, H.; STEIN, B.; ENGELS, G.: Constructing Efficient Information Extraction
           Pipelines. In: *20th ACM International Conference on Information and Knowledge Man-
           agement*. Ed. by BERENDT, B.; VRIES, A. de; FAN, W.; MACDONALD, C.; OUNIS, I.; RUTHVEN, I.
           ACM, Oct. 2011, pp. 2237–2240

[ZSG⁺20]   ZHANG, Y.; SUN, S.; GALLEY, M.; CHEN, Y.-C.; BROCKETT, C.; GAO, X.; GAO, J.; LIU, J.;
           DOLAN, B.: DialoGPT: Large-Scale Generative Pre-training for Conversational Response
           Generation. In: *Proceedings of the 58th Annual Meeting of the ACL*. Ed. by JURAFSKY, D.;
           CHAI, J.; SCHLUTER, N.; TETREAULT, J. ACL. 2020, pp. 270–278

Subproject B2:
Configuration and Evaluation

Jonas Hanselle[1], Eyke Hüllermeier[2], Felix Mohr[3], Axel Ngonga[1],
Mohamed Ahmed Sherif[1], Alexander Tornede[4], Marcel Wever[2]

1 Department of Computer Science, Paderborn University,
  Germany
2 Institute of Informatics, LMU Munich, Munich,
  Germany
3 Universidad de La Sabana, Chía, Colombia
4 Institute of Artificial Intelligence, Leibniz Unversity
  Hannover, Germany

Subproject B2 "Configuration and Evaluation" deals with methods and algorithms for the configuration and evaluation of software services in the OTF Computing scenario. During the three funding periods, various techniques have been developed and implemented for this purpose. Moreover, these techniques have been instantiated and evaluated on case studies from different domains: image processing, automated machine learning (AutoML), and question answering (QA) systems.

1   Introduction

Subproject B2 plays a central role within the CRC as a whole. Its task is to develop methods for the configuration of software services according to the requirement specifications provided by the user (cf. subproject B1). For this purpose, services traded on OTF markets are collected and assembled in an appropriate way. Before a service composition is executed, it will be analyzed for functional correctness (cf. subproject B3).

In the first period of the CRC, the focus of the subproject has been on

- the construction of a basic service configurator,

- the matching of services as an important part of the configuration process,

- the use of machine learning (ML) methods for the adptation of evaluation functions (in agreement with the users' preferences),

- the investigation of theoretical limits of the (automation of the) configuration process.

In the second period, subproject B2 concentrated on

- the extension of the configuration approach from a sequential to a sequential-hierarchical, template-based process,

jonas.hanselle@uni-paderborn.de (Jonas Hanselle), eyke@lmu.de (Eyke Hüllermeier), fe-lix.mohr@unisabana.edu.co (Felix Mohr), axel.ngonga@upb.de (Axel Ngonga), mohamed.sherif@uni-paderborn.de (Mohamed Sherif), a.tornede@ai.uni-hannover.de (Alexander Tornede), mar-cel.wever@ifi.lmu.de (Marcel Wever)

- the improvement of ML-based adaptation techniques by new methods from the field of preference learning,

- the adaptation of the configuration to market changes (e.g., the offering of new services or changing user preferences),

- a better interlinking of the configuration and the execution phase.

The third period was dedicated to

- the integration of the user in the configuration process, and the realization of this process in an online manner,

- the increase of the efficiency of automatic service configuration by exchanging information between different but related configuration processes,

- the use of quality criteria of services as part of the objective function,

- the broadening of the evaluation by means of a complementary case study in the field of question answering systems.

While the focus of the subproject was mainly on conceptual and methodological contributions, the development of methods and algorithms has been accompanied by concrete implementations from the very beginning. Moreover, conceptual solutions have been instantiated and evaluated on case studies from different domains, starting with image processing in the first funding period. Later on, the instantiation of service configuration has been realized for the practically relevant case of machine learning functionality, with the vision to establish "OTF Machine Learning" as an extension of what is currently known as "Automated Machine Learning" (AutoML) [HKV19]. In the last funding period, question answering systems have been added as a third application domain.

2   Highlights and Lessons Learned

In the following, we give an overview of the most important achievements of the subproject and summarize the key results that have been accomplished during the three funding periods of the CRC.

2.1   Domain-Independent Service Composition

During the first phase of the project, the main focus was on the functional aspect of automated service composition without a commitment to a specific domain. In this scenario, the user provides a formal specification of the functional requirements in terms of inputs, outputs, preconditions, and effects (IOPE) of the desired service. It is assumed that there is a set of existing services with the same type of descriptions that can be used to create the new desired service. The preconditions define types of and potential relationships between inputs. The effects describe conditions that the service guarantees to hold on the inputs or outputs after execution. The language used to describe preconditions and effects is a decidable subset of first-order logic. In particular, they serve to describe the *meaning* of outputs with respect to inputs. A simple example is a currency converter service that takes

a value *x* in EUR and returns its equivalent *y* in USD, and the effect could be described by *EUR2USD(x,y)*.

This task is an extended version of the classical planning problem. The services correspond to planning operators, instances of which can be connected into a chain of service calls, which correspond to actions in the classical planing setup. From this viewpoint, a service composition is a *plan*, and the initial state is the preconditions specified in the query, and the goal state is the effect specified in the query. The first crucial aspect that differentiates automated service composition from classical planning is that the operators can create new objects (the outputs), which is not supported in classical planning. A second difference is that the quality of a solution is not a scalar but a vector. In classical planning, the cost of a plan is the sum of the (scalar) costs of actions. However, in a service composition, several qualities of service (QoS) such as throughput, availability, privacy, etc. have to be considered in addition to the price.

Importantly, this type of problem is much more challenging than the much more commonly studied problem of pure QoS optimization over a *pre-defined* service workflow, where it is assumed that the general controller of the desired service is already implemented and connects to yet unspecified components through fixed interfaces. For each interface, a finite set of candidates is assumed to be available and can be plugged into the solution. The goal is then to pick the best combination of such candidates that, when put together, optimize the overall QoS of the configuration. This is a *configuration* of a controller, and the decisions the agent need to make to set up a solution are a strict subset of the decisions the agent has to make in the case of automated service composition.

Within this phase, we were the first to propose a planning algorithm that is capable of automatically creating service compositions based on functional descriptions in which the effects relate outputs to inputs while optimizing QoS. The approach is based on backward planning [MJB15] and is the first algorithm of its kind that is able to compose services not only based on the types of the inputs and outputs (monadic preconditions and effects), but can work with preconditions and effects of any arity. Starting from the goal definition with an empty composition, it tries to prepend service calls to the current composition that resolve at least one open requirement. Such a prepended service call typically comes with its own preconditions, which are added to the agenda unless they are provided in the initial state (preconditions warranted by the client in the query).

Another highlight of the approach is that it is able to prune nodes from the search space, if they are redundant in some way to make the search more efficient. First, it cuts nodes that encode compositions containing a service twice with the same inputs. Second, the algorithm prunes nodes of compositions that have a precondition that includes the precondition of one of its own subcompositions. That is, a node is pruned if it is associated with a composition whose preconditions are a superset of the preconditions of another composition.

In spite of its innovative aspects, the background search still suffered from a number of inefficiencies, which could be overcome by the development of a partial order planning algorithm [Moh]. The main advantage of a partial order planning (POP) compared to forward or backward planning is that it takes into account that the only constraint on the order of the planning operators is the flow of data. This specific property of the automated service composition problem implies that a huge number of serialized compositions are

equivalent from both the functional and the QoS viewpoint. This implies that the search space in forward or backward composition contains a huge number of mirrors, which are avoided in POP. In POP, orders in the plan are only partially fixed as far as *necessary*, based on the preconditions and effects of the operators used.

A further limitation of these compositions is that they cannot contain conditional paths let alone loops. To overcome this limitation, we proposed a template approach for loops in which a general structure with generic preconditions and effects is defined [MW15]. A replacement of service placeholders leads to a concrete service instantiation with concrete preconditions and effects. During a regular composition process, this mechanism can be invoked as a subroutine to create services with non-linear control flows on the fly.

All of the above composition approaches are based on orchestration. That means that it is assumed that there is a central instance that controls the service invocations and the data flow between them. In contrast to this, a choreography approach does not have a central controller, but all the participants of a composition are told about where, i.e., to which other services, they should send their output for a specific composition. This decentralized execution of service compositions can lead to enormous performance improvements, because the data has to travel very short distances (maybe even within the same compute center) compared to a centralized approach.

Based on the previous work, a choreography-based approach was developed in [JK16]. The main challenge in this approach is to trigger the execution of a service decentrally as soon as all the data of a service has arrived. In this work, the logic of service composition execution is modeled through Petri nets, in which data is seen as a resource and services as transitions that consume and produce data. Needless to say, the service does not actually consume the data, but the semantics of Petri nets are used to model the behavior of such a service.

In an alternative research thread in this phase, we investigated the issue that purely formal service descriptions are usually not suffecient to capture the user expectations. A common example for this is image processing. At the symbolic level, it is virtually impossible for the user to specify the desired transformation of an image. Instead, it can be sensible to show different proposals to the user and ask him for feedback. Such feedback can be binary, i.e., the user is rather satisfied than not with a result, or one provides several options and lets the user rank the alternatives. From this feedback, it is then in principle possible to learn which services (and their configurations) the user prefers over others. The main challenge is here to identify to which of the services within a composition to attribute a good or bad ranking.

To address this problem and to learn the relevance of a specific service (for a particular user in a concrete context), temporal difference (TD) learning was used [JM15]. We recognize that every service is, in a specific context, associated with a latent reward that is neither known before nor cannot be observed directly. However, if one interprets the set of partial compositions as the state space of an MDP, it is possible to learn the appropriateness of the components through TD learning. This is because TD learning propagates back the final evaluations to the state over time and, in this way, indirectly assigns ratings to partial compositions.

In this last approach, the composition technique deviates from the other techniques in that a forward search is adopted based on rules or tasks. The original problem is still to

convert an initial condition into a goal condition, but the services are no longer explicitly equipped with specific preconditions and effects, except maybe the types of the inputs and outputs. To decide whether a service is suitable for a specific task, the approach uses the concept of rules, which can be seen as possible ways of solving tasks. This view is closely related to hierarchical planning, which is also the basis of the ML-Plan approach developed in the second phase. The composition problem is then described through a context-free grammar in which the initial state is the start symbol, non-terminal symbols encode tasks, and production rules encode how tasks can be resolved. Such a rule maps a task to a series (usually of length 1) of services and possibly a new non-terminal. It can hence be seen as a task composition.

In summary, the first phase of the project focused on automated composition of services into a new service that satisfies the functional requirements specified by the user. To this end, classical planing was extended to support the generation of new planning objects and to support vector-valued QoS optimization. Based on the observation that fully ordered planning leads to significant inefficiencies, an alternative approach based on partial order planning was developed, that significantly outperforms the previously developed backward search. These orchestration-based approaches have been modified and extended in order to support choreography-based compositions, the latter of which were achieved by the means of Petri nets. Orthogonal to these efforts, we investigated the potential of composition approaches that take into account the fact that many important aspects of even the functional behavior of services cannot be captured in symbolic encodings. This makes it necessary to propose to the user a set of potentially satisfying solutions, all of which comply with the formal requirements posed by the user, and to ask the user for feedback.

The insights and experiences gained during this first phase were crucial for the definition of the goals in the following phases. One of the most important insights was indeed the limitation of formal service specifications. One very prominent example of automated service composition, where formal specifications are of no use is, automated machine learning. At the formal level, the goal here is simply to find a machine learning pipeline. The challenge is, however, that such pipelines work differently well on different datasets, and a pipeline that works well on one dataset may not work well on another one. Among hundreds of possible pipelines and billions of their configurations, the goal is to find the best suited one suited according to a performance measure such as accuracy.

2.2 ML-Plan: Configuring Machine Learning Pipelines

As mentioned in the previous section, a particularly interesting and practically relevant domain is machine learning. In this domain, services can process and model data in a wide variety of ways for different tasks. While there are many different functionally equivalent services for a task, the real interest is in finding services that satisfy certain non-functional properties, such as high accuracy and/or low prediction time. Since the non-functional properties can vary widely for different datasets and thus which service is best suited, it is important to determine the most appropriate service for each data set, which in turn requires expertise in the field of machine learning.

The need for applications with machine learning techniques has increased rapidly, especially in recent years, and cannot be satisfied by the available experts in this field. This

situation gave rise to the vision of automated machine learning (AutoML), which deals, among other things, with the automated selection and parameterization of machine learning algorithms. These algorithms are often arranged in a so-called pipeline where first the data is pre-processed and transformed in a certain way and eventually passed to a learning algorithm. Choosing the right algorithms also in the right order is of high importance to obtain the best possible results with respect to the non-functional requirements. In other words, AutoML deals with the automated and personalized delivery of machine learning applications.

Considering machine learning algorithms as services, the search for suitable machine learning algorithms or services fits seamlessly into the setting of OTF Computing, where a machine learning service or a composition of machine learning services on the requirements should be provided according to the user. While existing AutoML tools rely on techniques such as Bayesian optimization [THHL13; FKE+15], genetic programming [OBUM16; GV19] or reinforcement learning, we have continued our work with planning algorithms from the previous funding phase. More specifically, we have developed an AutoML system based on the paradigm of HTN-planning and using a best-first search for the search, which borrows concepts from the Monte Carlo tree search for the node evaluation.

Another problem is that with the ongoing search for suitable machine learning services, these adapt too much to the training data provided and do not generalize as well, which results in lower accuracy on new, unseen data. To avoid this effect, we propose a two-step AutoML process with ML-Plan [MWH18b], in which part of the training data is retained for a later final candidate selection. In a first phase, a pool of promising candidates can be put together with the reduced training data set and a final candidate can later be selected from this pool with the data that has not yet been used. In this way, the previously described effect, which is also referred to as overfitting in the literature, can be largely avoided.

In [MLHW18], we first developed an extension of HTN-planning to programmatic task network planning (PTN-planning), which can be used to combine the static search space model with dynamically determined ones. Information can be entangled to make the search space dependent on certain dynamic state properties. In addition, we compared ML-Plan with the state-of-the-art approaches and were able to determine a competitive performance for ML-Plan. A key component for the success of ML-Plan is the search space modeling based on HTN-planning, more specifically PTN-planning. The search space naturally exhibits hierarchical structures, e.g., learning algorithms that wrap other learning algorithms, for example, to tackle subproblem of the original problem. Furthermore, machine learning algorithms typically expose so-called hyperparameters which are parameters of the learning algorithm that may impact the learning behavior. Depending on which machine learning algorithm is chosen, different hyperparameters need to be optimized, introducing additional hierarchical structures and constraints. A schematic illustration of these hierarchical structures is shown in Figure 21 as well as in the following section (cf. Figure 23).

As already pointed out before, HTN-planning allows to capture those hierarchical structures and dependencies in a very natural way. In Figure 22 a search tree induced by HTN-planning and fast forward decomposition is shown, where an initial complex task is iteratively refined by other complex tasks or primitive task via so-called methods until only primitive tasks are left. The search space model follows a divide-and-conquer approach so that complex tasks are step-by-step broken down to (hopefully) simpler
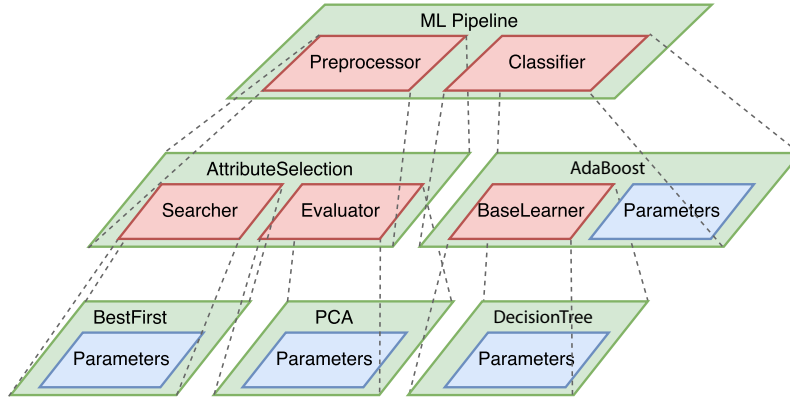
Figure 21: *A schematic illustration of a machine learning pipeline consisting of a prepro-
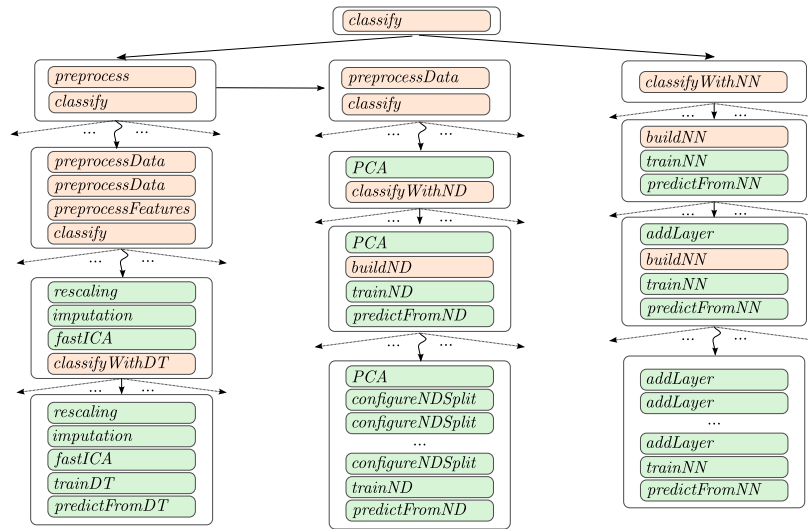cessing step and a learning algorithm.*

Figure 22: *Derivation of pipelines via hierarchical planning. Complex tasks are colored in
red and primitive tasks in green. Arcs indicate methods.*

tasks until all complex tasks have been refined by primitive tasks eventually. Intuitively
speaking, ML-Plan tries to imitate a human developer faced with a complex task of
providing a machine learning pipeline for a given problem. Then, this abstract task is
decomposed step-by-step to smaller abstract tasks such as choosing a learning algorithm
and preprocessing algorithms until all decisions regarding machine learning algorithms
and their hyperparameter values have been made.

ML-Plan has served as a starting point for several subsequent works. In a first sequel, we
extended the search space of ML-Plan from the commonly configured two-step pipelines,
involving a single pre-processing algorithm and a learning algorithm, to pipelines compris-
ing a potentially unlimited number of pre-processing algorithms arranged in a tree-shaped
structure and again a learning algorithm. In this way, ML-Plan is capable of building more
sophisticated data transformations to pre-process the given data. Furthermore, while ML-
Plan was originally developed for binary and multinomial classification tasks, it has been
extended to regression, multi-label classification [WMH18; WMTH19], and more recently,
remaining useful lifetime estimation in the realm of predictive maintenance [TTW+20].

Even in these settings, which are sometimes quite different from the original classification setting, it has shown strong performance, rendering ML-PLan a relatively flexible AutoML framework. To prepare ML-Plan for its deployment in an OTF market, where machine learning algorithms are provided in the form of cloud services which are computed in a distributed system, ML-Plan was also extended to work with services in a distributed environment [MWHF18; MWH18a].

Beyond scientific successes and academic publications, ML-Plan was a key component in the proof-of-concept project, a demonstrator joining various subprojects of the collaborative research center. More specifically, ML-Plan was used in the implementation of the on-the-fly provider for the configuration of the machine learning services. Therefore, ML-Plan represents the beating heart of one of the considered on-the-fly scenarios, i.e., on-the-fly machine learning [MWTH19].

2.3 Automated Configuration of Multi-Label Classifiers

Another relevant learning problem, which is also extremely interesting from an AutoML point of view, is the so-called multi-label classification. Here, in contrast to conventional, single label, classification problems (SLC), instances can be associated not only with one class, but with several classes at the same time. Consequently, instead of mapping from $\mathcal{X}$ to $\mathcal{L}$, where $\mathcal{X}$ is the instance space and $\mathcal{L}$ is the set of class labels, models map to the power set of $\mathcal{L}$, i.e., all possible label combinations. While single-label classification tries to learn primarily dependencies between $\mathcal{X}$ and $\mathcal{L}$, much of the MLC literature also tries to exploit dependencies between labels, i.e., between elements in $\mathcal{L}$, in order to increase the generalization goodness.

Based on methods for SLC, a diverse repertoire of MLC methods has been developed over time. One strain of the literature adapts SLC models and/or learning algorithms for the MLC setting, so-called algorithm adaptation approaches. Alternatively, MLC problems are transformed into one or multiple SLC problem(s) such that in turn already well-studied SLC methods can be applied to the induced problems.

An exemplary selection of algorithms constituting a multi-label classifier is illustrated in Figure 23.

From an AutoML perspective, problem transformation methods need to be configured with an SLC method as a base learner, and the choice of both the problem transformation method as well as the baselearner depends on the task in question, i.e., the dataset and loss function. Hence, the search space for automatically selecting algorithms and optimizing their hyperparameters is a multiple of the search space of SLC, which is reported already huge, since the search space for SLC is included for each MLC problem transformation method. Also in this direction of extending AutoML methods it is questionable to what extent the already proposed methods can be applied to the MLC setting. More precisely, questions of scalability arise.

Another question is how to design a fair and meaninigful comparison. In the literature, oftentimes complete AutoML systems are proposed that combine an optimization method with a custom search space definition and a module for evaluating solution candidates. However, we are interested in how well optimization methods scale with the increasing
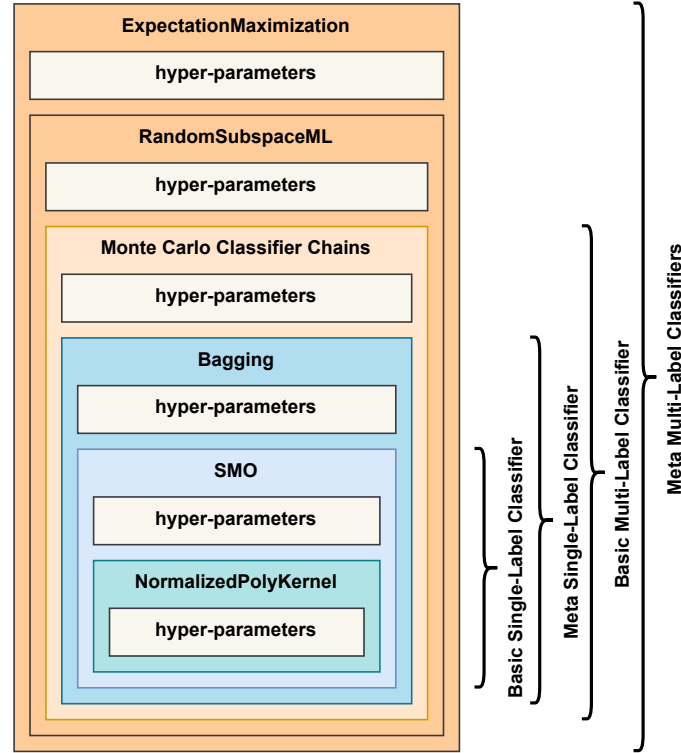
Figure 23: *A schematic illustration of the structure of a multi-label classifier following a problem transformation strategy. Such a multi-label classifier may comprise multiple "layers" of algorithms where for each layer one can in principle choose between different algorithms of that type.*

search space size, and how well they can handle the AutoML for MLC problem. To be clear, we are not interested in an AutoML system as such, but to investigate which optimization method is best suited for this particular setting. This requires to unify certain design decisions as for instance the search space and the evaluation module, as well as other more technical design decisions: parallelization, degree of parallelization, memory constraints, etc. Moreover, AutoML systems often work with different ML libraries as a backend, i.e., some systems may work with scikit-learn [PVG⁺11], some with WEKA [HFH⁺09], and again others may work with mlr3 [LBR⁺19]. However, implementations of the same methods may differ significantly and oftentimes some methods are not even available in all libraries. Thus, for a fair comparison of optimization methods they should be benchmarked in a unified environment such that all the optimization methods use exactly the same implementations of the solution candidate evaluation and operate on the same search space. We interpret the latter in a way that all optimization methods may potentially encounter every candidate another optimization method may be able to find. Of course, the precise specification of the search space may differ from optimization method to optimization method.

In [WTMH21] we present answers to both questions: How to benchmark different optimization methods proposed for AutoML in the SLC setting and to what extent those methods appear to scale well with the specifics of the MLC setting. To this end, we first propose a benchmarking framework which, in principle, can be used to benchmark any type of combined algorithm selection and hyperparameter optimization problem setting.

Meaning the benchmark is not limited to MLC problems but can be used for SLC problems as well. The implementation of the benchmark framework is cross-platform and can be used to integrate, for example, implementations in Python and Java. This allows optimization methods implemented for different platforms to work with the same evaluation module without the need for re-implementing the optimization method for another platform. Beside the software, the benchmark also comes with constraints on the hardware to use and a limit on the time budget. Regarding the time budget the proposed benchmark restricts both the total runtime and the runtime for evaluating a solution candidate. The latter is an important aspect since the time allowed for evaluating a single solution candidate implicitly prunes slower candidates from the search space and thus different approaches would again operate on different search spaces.

Based on this benchmark framework, an extensive empirical study was conducted comparing 6 optimization methods for a total of 24 datasets with 10 different train-test splits each. Furthermore, we considered a total of 3 performance measures for optimization, which generalize the F1 measure in three different ways from the SLC to the MLC setting. Generally speaking, we found that all the six optimization methods are struggling with the MLC setting, taking quite some time to return reasonable solutions. In fact, on average, only after 4 hours the optimization methods reach a level which is at least close to the best result that can be obtained after 24h. Most interestingly, we find that rather greedy approaches such as the optimization method employed in our AutoML system ML-Plan and Hyperband, an optimization method based on the successive halving paradigm, appear to perform overall best. We hypothesize that this is due to the fact that the choice of the algorithm is more important than tuning the hyperparameters. Furthermore, the evaluation of solution candidates is typically more costly so that multi-fidelity optimization appears to be indeed a crucial characteristic for an optimization method aiming to automate multi-label classification.

2.4 Censored Data in Algorithm Selection

Algorithm selection (AS) [Ric76; KHNT19] is the task of finding the most suitable algorithm for a given problem instance of an algorithmic problem domain, such as the Boolean satisfiability problem (SAT) or classification (machine learning). Typically, suitability is measured in terms of a performance measure, which characterizes some sort of solution quality which shall be maximized or some kind of cost which shall be minimized. One of the most prominent performance measures is the runtime an algorithm needs in order to return a valid solution, which is of special interest in the domain of hard combinatorial problems such as SAT or integer optimization. A common approach towards per-instance algorithm selection is the use of machine learning, in which runtime measurements of algorithms from previous runs are used in order to estimate the algorithms' runtimes on new, previously unseen problem instances.

AS is of particular interest for the CRC, as it is a subproblem of AutoML that can be solved with conceptually simpler approaches making it easier to study certain properties associated with it. In particular, in this section, we elaborate on the problem of so-called right-censored training data [KK10], which can be found quite frequently in AS, algorithm configuration (AC) [SBT+22], hyperparameter optimization (HPO) [FH19; BBL+21] and AutoML problems.

In order to deduce estimators for the algorithms' runtimes, one generally assumes that problem instances can be represented in terms of characteristics, so called features or meta-features in the context of meta-learning [Van18], that should be correlated with the performance measure, in this case runtime. Correspondingly, the training data needed to learn such estimators consists of feature descriptions of problem instances and the runtimes achieved by various algorithms on this particular problem instance. Since algorithms for such hard problems may exhibit extremely long runtimes, they are generally not run for an indefinite amount of time until they eventually terminate, but are rather terminated externally once the time exceeds a certain threshold T, called cutoff. Thus, the training data contains right-censored datapoints, i.e. observations of which we do not know the exact runtime, but only a lower bound T.

Naturally, such a right-censored datapoint is not a scalar value, but rather a right-open interval and thus cannot be used as a standard regression training datapoint, but has to be treated differently. The community has suggested a variety of approaches in the literature of how to handle such datapoints in the context of AS, AC, HPO and AutoML [XHHL07; HTWH20; HTWH21; HHL11; ELH+18; EHM+20].

The simplest approach for dealing with such censored samples is to ignore them all together, which, however, comes with a loss of information. Although the censored data cannot be used directly as training points, they do contain information, which should be incorporated. Another simple strategy is imputation. For example, in the case when algorithm selectors are evaluated based on the so-called *PAR10* score [KHNT19] - a penalized version of runtime - censored samples are commonly replaced by the cutoff time T or a multiple thereof, such as $10\,T$. Obviously, such imputations can easily result in a strong bias of the model learned on such data [Gre05]. A more sophisticated approach to impute right-censored data developed by [SH79] samples from a truncated normal distribution and is leveraged by many AS and AC approaches (e.g. [XHHL07; ELH+18]). However, as we show in [TWW+20b], these approaches do not necessarily improve upon the naive imputation schemes discussed above.

All of the approaches presented above share the problem that they are rather indirect solutions for dealing with censored data and as such, come with the disadvantages noted above. In contrast to that, methods from the field of survival analysis [KK10] (SA) can inherently deal censored datapoints and are thus much more suited for the kind training data often found in AS, AC, HPO and AutoML problems. Correspondingly, in [TWW+20a] we adapt rigorous statistical SA methods for constructing algorithm selectors using partially right-censored runtime data. In particular, using random survival forests [IKBL08], we learn algorithm runtime distributions, which we then leverage to obtain an estimated algorithm runtime.

In order to derive a point estimate of the runtime of an algorithm from the runtime distribution, a first natural choice is the expectation of the distribution, i.e., the expected algorithm runtime. While this does indeed yield reasonably good algorithm selections in many practical cases, the expected value can be overly optimistic for the selection of an algorithm, if the performance measure penalizes timeouts of algorithms excessively as is the case for the *PAR10* score.

To mitigate overly optimistic selections in such cases, we advocate for a decision-theoretic selection approach that incorporates the concept of risk aversion, which coincides with
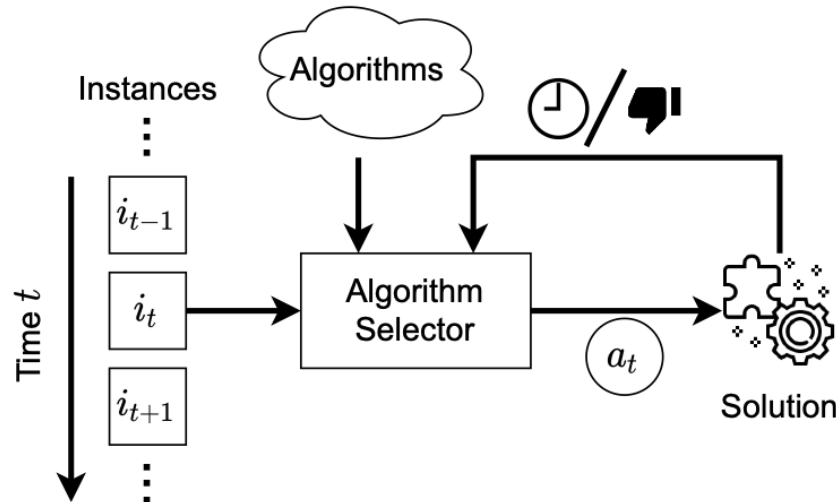
Figure 24: *General process of the online algorithm selection setting. In each round, the selector is asked to select an algorithm, which is then evaluated using the performance measure resulting in an evaluation result fed back to the learner. Based on this result, the learner can update its internal model.*

timeout aversion in our case. To this end, we compute the expectation of a risk-averse loss function applied to the random variable modeling the runtime of an algorithm instead of directly computing the expectation of that random variable.

Combining the concepts of SA and decision-theoretic risk aversion allows us to achieve state-of-the-art algorithm selection performance on the de-facto standard AS benchmark, called ASLib [BKK+16], beating the hitherto state of the art by roughly 15%.

Standard AS considers an offline problem in the sense that one usually assumes a phase prior to the actual application of the selector, where any form of data generation and learning can take place. In contrast, online AS (OAS) weakens this assumptions and instead aims at selectors, which are learned and updated online in a round-wise manner without any prior learning phase. For this purpose, in each round, the selector is asked to select an algorithm, which is then evaluated using the performance measure, resulting in an evaluation result fed back to the learner. Based on this result, the learner can update its internal model. The process is depicted in Figure 24.

The problems associated with censored data are even more prominent in the online case, as one only obtains a single datapoint each round, which might even be censored. If censored datapoints are, for example, dropped in such cases instead of incorporated into the learning process, the model cannot be updated and no learning takes place for that round.

Unfortunately, the SA methods we previously discussed cannot be used in the online setting, as the vast majority of such methods are inherently designed as offline approaches. For example, Cox' proportional hazards model [Cox72] leverages the Breslow estimator to estimate the baseline survival function, which has to store all data previously seen in the form of risk-sets [Bre72]. Naturally, storing all previously seen data is not a viable approach in an online setting as the storage complexity grows with the time horizon in such a case.

As an alternative solution, in [TBH22] we suggest to adapt well-known bandit algorithms to OAS and runtime-oriented loss functions. In particular, we investigate the bias incurred by directly applying a UCB strategy [ACF02], when censored samples are dropped completely or imputed with the cutoff T. The corresponding bias-correction terms result in extremely large confidence bounds that do no longer yield reasonable algorithm selections in practice. To alleviate these problems, we propose a Thompson sampling approach [Tho33; RRK$^+$18] that is adapted to losses strongly penalizing algorithm timeouts and imputes censored samples by an online variant of the Schmee&Hahn approach [BJ79].

With this approach we can improve upon existing OAS approaches in terms of selection performance while featuring a runtime and space complexity independent of the time horizon - a property that other existing approaches do not offer.

2.5   ADAGIO - Automated Data Augmentation of Knowledge Graphs Using Multi-Expression Learning

The creation of an RDF knowledge graph for a particular application commonly involves a pipeline of tools that transform a set of input data sources into an RDF knowledge graph in a process called dataset augmentation. The components of such augmentation pipelines often require extensive configuration to lead to satisfactory results. Thus, non-experts are often unable to use them. In this section, we present the basic idea behind ADAGIO [DSN22], an efficient supervised algorithm based on genetic programming for learning knowledge graph augmentation pipelines of arbitrary length. Our approach uses multi-expression learning to learn augmentation pipelines able to achieve a high F-measure on the training data. Our evaluation suggests that our approach can efficiently learn a larger class of RDF dataset augmentation tasks than the state of the art while using only a single training example. Even on the most complex augmentation problem we posed, our approach consistently achieves an average F_1-measure of 99% in under 500 iterations with an average runtime of 16 seconds.

RDF Dataset. An *RDF* dataset D is a set of triples $\{(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})\}$, where $\mathcal{R}$ is the set of all RDF IRI resources, $\mathcal{B}$ is the set of all RDF blank nodes and $\mathcal{L}$ is the set of all RDF literals. We denote the set of all RDF datasets as $\mathcal{D}$.

Dataset Operators. A function $\mathbb{O}_{(n,m)}\colon \mathcal{D}^{n+1} \to \mathcal{D}^m$ is called a *dataset operator*. Intuitively, a dataset operator $\mathbb{O}_{(n,m)}$ processes n input datasets using another dataset C as configuration to produce m output datasets. We call n the in-degree and m the out-degree of $\mathbb{O}_{(n,m)}$ and will resort to writing just $\mathbb{O}$ when the lack of their specification will incur no loss of generality. Given integers $i \in [1, n]$, $j \in [1, m]$, we call the ith argument of $\mathbb{O}_{(n,m)}$ and the jth component in the output of $\mathbb{O}_{(n,m)}$ the in-port i and out-port j, respectively. The set of all dataset operators is denoted as $\mathbb{O}$.

Augmentation Graphs.   An augmentation graph $G = (\mathbf{O}, \mathbf{E}, \mathbf{L}, \mathbf{M})$ is a directed acyclic labeled multigraph where $\mathbf{O}$ is a set of dataset operators, which act as vertices; $\mathbf{E}$ is the set of edges, which represent flow of data; $\mathbf{L}$ is the edge labeling function, which defines mappings between dataset operator out-ports and in-ports for a given edge; and $\mathbf{M}$ is a mapping from vertices to configuration datasets. We call the subsets of vertices with 0 in-degree *root vertices*. *leaf vertices* are the and subsets of vertices with 0 out-degree.
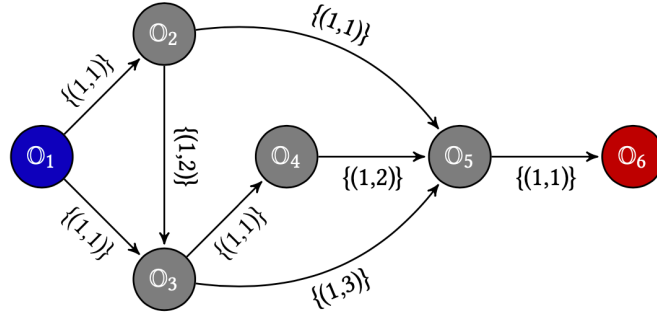
Figure 25: *Running example augmentation graph.*

All other vertices are *inner vertices*. Note that, per definition all root vertices of an augmentation graph must be dataset emitters, all leaf vertices must be dataset acceptors and all inner vertices must be augmentation operators. In our running example augmentation graph in Figure 25, we coloured all root vertices blue and all leaf vertices red. The intuition behind $\mathbf{L}$ is that, given $e = (\mathbb{O}_1, \mathbb{O}_2)$, we need to define which of $\mathbb{O}_1$'s out-ports map to which of $\mathbb{O}_2$'s in-ports. For instance, in our running example in Figure 25, the label set on the edge between $\mathbb{O}_4$ and $\mathbb{O}_5$ indicates that $\mathbb{O}_4$'s first output dataset is the second argument to $\mathbb{O}_5$. To evaluate an augmentation graph, we first obtain the RDF datasets as output of the root vertices in $\mathbf{O}_r$. These datasets then flow through the graph as specified by the semantics we associated with the edge set $\mathbf{E}$ and the label multiset $\mathbf{L}$. Whenever a dataset operator $\mathbb{O}_{(n,m)} \in \mathbf{O}_i$ has received all its n input datasets, it is evaluated using $\mathbf{M}(\mathbb{O}_{(n,m)})$ as its last argument. The flow through the graph continues until eventually all vertices have been evaluated. We call an augmentation graph G *linear* if there exists at most a single path between $\mathbb{O}_1$ and $\mathbb{O}_2$; *semi-linear* if there is a pair of vertices $u, v \in \mathbf{O}$, $u \neq v$ for which there exist multiple paths from u to v; *confluent* if it has multiple root vertices and exactly one leaf vertex; *inherently confluent* if it is confluent and it only contains confluent augmentation operators, *general* otherwise.

Augmentation Tables.   An augmentation table $\mathbb{T}$ is a condensed linear representation for inherently confluent augmentation graphs based on column tables [KP98]. The idea behind this representation is that each row represents one dataset operator. We can go through this table from top to bottom and evaluate the dataset operators which correspond to a row i using only the results of rows 1 to $i-1$. Since dataset acceptors produce no output, they are omitted in this representation for the sake of simplicity.

Let $G = (\mathbf{O}, \mathbf{E}, \mathbf{L}, \mathbf{M})$ be an inherently confluent augmentation graph. Moreover, let $N(\mathbf{O}) := \max \{n \mid \mathbb{O}_{(n,m)} \in O\}$ denote *the maximum in-degree* in $\mathbf{O}$. An augmentation table is a table with $3 + N(\mathbf{O})$ columns and $|\mathbf{O}|$ rows, where the first column contains dataset operators, the second column contains configuration datasets and the third column contains the in-degrees of the dataset operators in the first column. The last $N(\mathbf{O})$ columns contain the indices of the rows used as input to the corresponding dataset operator. Given an augmentation table $\mathbb{T}$, we write $\mathbb{T}_i$ and $\mathbb{T}_{i,j}$ to refer to the ith row and the jth column in the ith row of $\mathbb{T}$, respectively. Applying this representation to our running example augmentation graph in Figure 25 gives the augmentation table depicted in Table 1. The algorithm for the computation of an augmentation table from a given inherently confluent augmentation graph is given in [KP98].

Table 1: *Running example augmentation table.*

$\mathbb{T}_1$:	$\mathbb{O}_1$	C_1	0	0	0	0
$\mathbb{T}_2$:	$\mathbb{O}_2$	C_2	1	1	0	0
$\mathbb{T}_3$:	$\mathbb{O}_3$	C_3	2	1	2	0
$\mathbb{T}_4$:	$\mathbb{O}_4$	C_4	1	3	0	0
$\mathbb{T}_5$:	$\mathbb{O}_5$	C_5	3	2	4	3

We call a row within an augmentation table an *output row*, if it is not used as input to a subsequent row. Note that output rows always correspond to dataset acceptors and that our previous definition of augmentation tables allows for only a single output row, as our augmentation tables must be isomorphic to inherently confluent augmentation graphs.

Multi-Expressive Augmentation Tables. A *multi-expressive augmentation table* is a generalized augmentation table that has more than one *output row*. Note that, any row in a multi-expressive augmentation table can be seen as an output row by just disregarding all rows below it. Given such a *reference output row* in a multi-expressive augmentation table, we can derive a normal augmentation table by following the procedure introduced in [DSN22].

Problem Definition. The problem under study is to find an adequate enrichment graph for a given training example. We restrict ourselves to learning the subclass of inherently confluent enrichment graphs. We will furthermore restrict our study to enrichment graphs where the maximum in-degree of the involved enrichment operators and the number of involved dataset emitters are at most two.

Learning Algorithm. The core of our learning approach is a population-based $(\mu + \lambda)$ *multi-expression learning* (MEP) algorithm[10] that is able to learn the subclass of inherently confluent augmentation graphs. Our population consists of a fixed number $\mu + \lambda$ of multi-expressive augmentation tables that we also call *genotypes*. All genotypes have a fixed number r of rows. Tournament selection [MG95] with a tournament size of 3 and a selection probability of 0.75 is applied for determining the mating pool and for selecting the survivors. We use 1-elitist selection [Mit98] to avoid a decrease in fitness. Both the offspring and the survivors are subject to mutation. The *offspring fraction* $\alpha = \frac{\mu}{\lambda}$, *mutation probability* σ and *mutation rate* ρ are hyperparameters that need to be determined experimentally. Therefore, we ran a series of grid searches on augmentation tasks with increasing difficulty and used our insights from previous runs to fine-tune the next. We report the final grid search results in Figure 26. These results suggest that the best set of hyperparameters are the *offspring fraction* $\alpha = 1$, the *mutation probability* $\sigma = 0.5$ and the *mutation rate* $\rho = 0.5$.

The algorithm will stop when either a perfect solution is found, a maximum number g of generations is exceeded or our convergence detection terminates it. As the results of RDF dataset augmentation are commonly expected to have a regular structure, we can expect the output dataset to be decomposable into a number of subgraphs that are isomorphic up to a certain error w.r.t. some structural graph similarity measure. We therefore regard the training examples as a list of source *concise bounded descriptions* (CBDs) and a single target CBD of sufficient depth to representatively capture the desired augmentation. This is

---

[10]μ is the *population size* and λ is the *recombination pool size*.
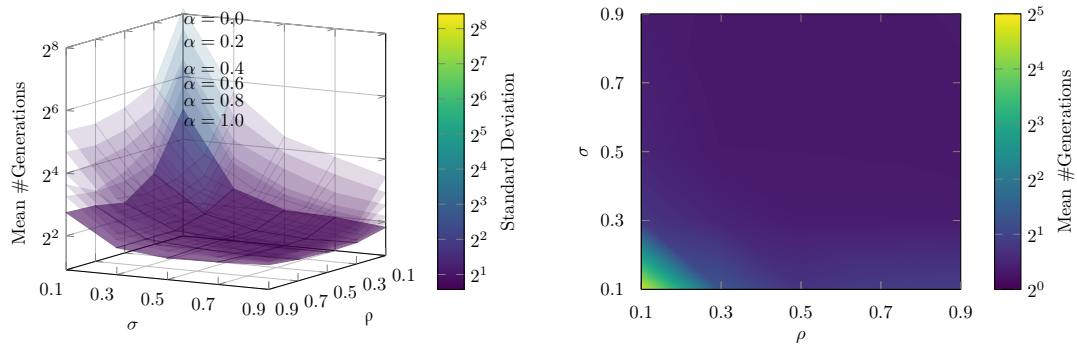
Figure 26: *Hyperparameter Optimization Results.*

in accordance with the observation that a single pair of CBD often suffices for the training of augmentation pipelines [SNL15]. Note our choice to restrict the number of involved dataset emitters to at most two.

3   Conclusion and Outlook

Subproject B2 plays a central role within the overall architecture of the CRC, since the automatic configuration of software services is at the core of the OTF Computing paradigm. As such, it is closely connected to other subprojects, which either build on the service configurations provided by B2 (such as B3, which is responsible for the formal verification of configurations), or provide important input (most notably the service specifications produced by B1).

Starting with a relatively abstract, logic-based approach using formal specifications of functional requirements and techniques from automated planning for service composition, the focus of this subproject has shifted toward more concrete applications, such as automated machine learning (AutoML) and query answering (QA), and the use of data-driven methods for service composition. Interestingly, this has led to attributing a double role to machine learning, which served as a key methodology for automated service composition and, simultaneously, as an important use case.

Tackling the problem of automated service configuration for more concrete applications was mainly motivated by the observation that developing methods for this task, including the formal specification of requirements with preconditions and effects, the formalization of underlying domain knowledge, etc., is very difficult and hardly practicable on a completely generic level. Moreover, many criteria influencing the quality of a service, and hence being essential for the optimization of a composition, cannot be assessed in a purely formal way. Instead, a service composition must be realized and executed — for example, the quality of a machine learning pipeline can only be judged on an implementation level, by running it and applying it to a real data set.

The research conducted in the course of this subproject has not only contributed to the OTF framework of the CRC, but also created impact in other fields and scientific disciplines. A notable example is our work on AutoML, most visibly manifested in the AutoML

tool ML-Plan, which has been well received by the research community. In spite of this success, the vision we have for this field has not yet been realized: Going beyond the use of individual tools for AutoML, we envision "OTF Machine Learning" as a natural next step in the evolution of AI technology, and an important contribution to the democratization of AI. What we mean by OTF-ML is the realization of the OTF computing paradigm for the specific case of machine learning (or, more generally, data science) functionality, not restricted to individual software tools but including the entire compute and market infrastructure. We are convinced that this vision will become reality in the not too distant future, also thanks to the foundations that have been laid by this CRC.

Bibliography

[ACF02]   Auer, P.; Cesa-Bianchi, N.; Fischer, P.: Finite-time analysis of the multiarmed bandit problem. In: *Machine Learning* 47 (2002), no. 2-3, pp. 235–256.

[BBL+21]   Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L., et al.: Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. In: *arXiv preprint arXiv:2107.05847* (2021)

[BJ79]   Buckley, J.; James, I.: Linear regression with censored data. In: *Biometrika* 66 (1979), no. 3, pp. 429–436

[BKK+16]   Bischl, B.; Kerschke, P.; Kotthoff, L.; Lindauer, M.; Malitsky, Y.; Fréchette, A.; Hoos, H. H.; Hutter, F.; Leyton-Brown, K.; Tierney, K.; Vanschoren, J.: ASlib: A benchmark library for algorithm selection. In: *Artif. Intell.* 237 (2016)

[Bre72]   Breslow, N. E.: Contribution to discussion of paper by DR Cox. In: *Journal of the Royal Statistical Society* 34 (1972), pp. 216–217

[Cox72]   Cox, D. R.: Regression models and life tables (with discussion). In: *Journal of the Royal Statistical Society* 34 (1972), no. 2, pp. 187–220

[DSN22]   Dressler, K.; Sherif, M. A.; Ngomo, A.-C. N.: ADAGIO - Automated Data Augmentation of Knowledge Graphs Using Multi-expression Learning. In: *Proceedings of the 33rd ACM Conference on Hypertext and Hypermedia*. 2022.

[EHM+20]   Eggensperger, K.; Haase, K.; Müller, P.; Lindauer, M.; Hutter, F.: Neural model-based optimization with right-censored observations. In: *CoRR abs/2009.13828 (2020)*. arXiv: `2009.13828`.

[ELH+18]   Eggensperger, K.; Lindauer, M.; Hoos, H. H.; Hutter, F.; Leyton-Brown, K.: Efficient benchmarking of algorithm configurators via model-based surrogates. In: *Machine Learning* 107 (2018), no. 1, pp. 15–41.

[FH19]   Feurer, M.; Hutter, F.: Hyperparameter optimization. In: *Automated machine learning*. Springer, Cham, 2019, pp. 3–33

[FKE+15]   Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J. T.; Blum, M.; Hutter, F.: Efficient and Robust Automated Machine Learning. In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; Garnett, R. 2015, pp. 2962–2970.

[Gre05]   Greene, W. H.: Censored data and truncated distributions. In: *NYU Working Paper* (2005)

[GV19]   Gijsbers, P.; Vanschoren, J.: GAMA: Genetic Automated Machine learning Assistant. In: *J. Open Source Softw.* 4 (2019), no. 33, p. 1132.

[HFH+09]   Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H.: The WEKA data mining software: an update. In: *ACM SIGKDD explorations newsletter* 11 (2009), no. 1, pp. 10–18

[HHL11]   HUTTER, F.; HOLGER H. HOOS; LEYTON-BROWN, K.: Bayesian optimization with censored
             response data. In: *NIPS workshop on Bayesian Optimization, Sequential Experimental
             Design and Bandits*. Dec. 2011

[HKV19]   HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J., eds.: *Automated Machine Learning - Methods,
             Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer,
             2019.

[HTWH20]   HANSELLE, J.; TORNEDE, A.; WEVER, M.; HÜLLERMEIER, E.: Hybrid ranking and regression
             for algorithm selection. In: *KI 2020: Advances in Artificial Intelligence - 43rd German
             Conference on AI*. 2020, pp. 59–72.

[HTWH21]   HANSELLE, J.; TORNEDE, A.; WEVER, M.; HÜLLERMEIER, E.: Algorithm selection as superset
             learning: Constructing algorithm selectors from imprecise performance data. In: *Advances
             in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021*.
             2021, pp. 152–163.

[IKBL08]   ISHWARAN, H.; KOGALUR, U. B.; BLACKSTONE, E. H.; LAUER, M. S.: Random survival forests.
             In: *The annals of applied statistics* 2 (2008), no. 3, pp. 841–860

[JK16]   JUNGMANN, A.; KLEINJOHANN, B.: Automatic Composition of Service-Based Image Process-
             ing Applications. In: *2016 IEEE International Conference on Services Computing (SCC)*.
             2016, pp. 106–113

[JM15]   JUNGMANN, A.; MOHR, F.: An approach towards adaptive service composition in markets
             of composed services. In: *Journal of Internet Services and Applications* 6 (2015), no. 1,
             pp. 1–18

[KHNT19]   KERSCHKE, P.; HOOS, H. H.; NEUMANN, F.; TRAUTMANN, H.: Automated algorithm selection:
             Survey and perspectives. In: *Evolutionary Computation* 27 (2019), no. 1, pp. 3–45.

[KK10]   KLEINBAUM, D. G.; KLEIN, M.: *Survival Analysis*. Vol. 3. Springer, 2010

[KP98]   KVASNIÈKA, V.; POSPÍCHAL, J.: Simple Implementation of Genetic Programming by Column
             Tables. In: *Soft Computing in Engineering Design and Manufacturing*. Ed. by CHAWDHRY,
             P. K.; ROY, R.; PANT, R. K. London: Springer London, 1998, pp. 48–56.

[LBR+19]   LANG, M.; BINDER, M.; RICHTER, J.; SCHRATZ, P.; PFISTERER, F.; COORS, S.; AU, Q.; CASAL-
             ICCHIO, G.; KOTTHOFF, L.; BISCHL, B.: mlr3: A modern object-oriented machine learning
             framework in R. In: *J. Open Source Softw.* 4 (2019), no. 44, p. 1903.

[MG95]   MILLER, B. L.; GOLDBERG, D. E.: Genetic Algorithms, Tournament Selection, and the
             Effects of Noise. In: *Complex Systems* 9 (1995), no. 3.

[Mit98]   MITCHELL, M.: *An introduction to genetic algorithms*. MIT Press, 1998.

[MJB15]   MOHR, F.; JUNGMANN, A.; BÜNING, H. K.: Automated Online Service Composition. In: *2015
             IEEE International Conference on Services Computing*. 2015, pp. 57–64

[MLHW18]   MOHR, F.; LETTMANN, T.; HÜLLERMEIER, E.; WEVER, M.: Programmatic task network plan-
             ning. In: *Proceedings of the 1st ICAPS Workshop on Hierarchical Planning*. 2018, pp. 31–
             39

[Moh]   MOHR, F.: Towards automated service composition under quality constraints. PhD thesis.
             Dissertation, Paderborn, Universität Paderborn, 2016

[MW15]   MOHR, F.; WALTHER, S.: Template-based generation of semantic services. In: *International
             Conference on Software Reuse*. Springer. 2015, pp. 188–203

[MWH18a]   MOHR, F.; WEVER, M.; HÜLLERMEIER, E.: Automated Machine Learning Service Composi-
             tion. In: *CoRR abs/1809.00486* (2018). arXiv: `1809.00486`.

[MWH18b]   MOHR, F.; WEVER, M.; HÜLLERMEIER, E.: ML-Plan: Automated machine learning via
             hierarchical planning. In: *Mach. Learn.* 107 (2018), no. 8-10, pp. 1495–1515

[MWHF18]   MOHR, F.; WEVER, M.; HÜLLERMEIER, E.; FAEZ, A.: (WIP) Towards the Automated Compo-
             sition of Machine Learning Services. In: *2018 IEEE International Conference on Services
             Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE, 2018, pp. 241–244

[MWTH19]   MOHR, F.; WEVER, M.; TORNEDE, A.; HÜLLERMEIER, E.: From Automated to On-The-Fly Machine Learning. In: *LNI P-294* (2019), pp. 273–274

[OBUM16]   OLSON, R. S.; BARTLEY, N.; URBANOWICZ, R. J.; MOORE, J. H.: Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016.* Ed. by FRIEDRICH, T.; NEUMANN, F.; SUTTON, A. M. ACM, 2016, pp. 485–492.

[PVG+11]   PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V., et al.: Scikit-learn: Machine learning in Python. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830

[Ric76]   RICE, J. R.: The Algorithm Selection Problem. In: *Adv. Comput.* 15 (1976), pp. 65–118.

[RRK+18]   RUSSO, D.; ROY, B. V.; KAZEROUNI, A.; OSBAND, I.; WEN, Z.: A tutorial on Thompson sampling. In: *Foundations and Trends in Machine Learning* 11 (2018), no. 1, pp. 1–96.

[SBT+22]   SCHEDE, E.; BRANDT, J.; TORNEDE, A.; WEVER, M.; BENGS, V.; HÜLLERMEIER, E.; TIERNEY, K.: A Survey of Methods for Automated Algorithm Configuration. In: *Journal of Artificial Intelligence* (2022)

[SH79]   SCHMEE, J.; HAHN, G. J.: A simple method for regression analysis with censored data. In: *Technometrics* 21 (1979), no. 4

[SNL15]   SHERIF, M. A.; NGOMO, A.-C. N.; LEHMANN, J.: Automating RDF Dataset Transformation and Enrichment. In: *The Semantic Web. Latest Advances and New Domains*. Ed. by GANDON, F.; SABOU, M.; SACK, H.; D'AMATO, C.; CUDRÉ-MAUROUX, P.; ZIMMERMANN, A. Cham: Springer International Publishing, 2015, pp. 371–387

[TBH22]   TORNEDE, A.; BENGS, V.; HÜLLERMEIER, E.: Machine Learning for Online Algorithm Selection under Censored Feedback. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (June 2022), no. 9. Number: 9, pp. 10370–10380.

[THHL13]   THORNTON, C.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. Ed. by DHILLON, I. S.; KOREN, Y.; GHANI, R.; SENATOR, T. E.; BRADLEY, P.; PAREKH, R.; HE, J.; GROSSMAN, R. L.; UTHURUSAMY, R. ACM, 2013, pp. 847–855.

[Tho33]   THOMPSON, W. R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. In: *Biometrika* 25 (1933), no. 3/4, pp. 285–294

[TTW+20]   TORNEDE, T.; TORNEDE, A.; WEVER, M.; MOHR, F.; HÜLLERMEIER, E.: AutoML for Predictive Maintenance: One Tool to RUL Them All. In: *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning - Second International Workshop, IoT Streams 2020, and First International Workshop, ITEM 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14-18, 2020, Revised Selected Papers*. Ed. by GAMA, J.; PASHAMI, S.; BIFET, A.; MOUCHAWEH, M. S.; FRÖNING, H.; PERNKOPF, F.; SCHIELE, G.; BLOTT, M. Vol. 1325. Communications in Computer and Information Science. Springer, 2020, pp. 106–118.

[TWW+20a]   TORNEDE, A.; WEVER, M.; WERNER, S.; MOHR, F.; HÜLLERMEIER, E.: Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis. en. In: *Proceedings of The 12th Asian Conference on Machine Learning*. ISSN: 2640-3498. PMLR, Sept. 2020, pp. 737–752.

[TWW+20b]   TORNEDE, A.; WEVER, M.; WERNER, S.; MOHR, F.; HÜLLERMEIER, E.: Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis. In: *Proceedings of The 12th Asian Conference on Machine Learning, ACML 2020, 18-20 November 2020, Bangkok, Thailand*. Vol. 129. Proceedings of Machine Learning Research. PMLR, 2020, pp. 737–752.

[Van18]   VANSCHOREN, J.: Meta-learning: A survey. In: *arXiv preprint arXiv:1810.03548* (2018)

[WMH18]   Wever, M. D.; Mohr, F.; Hüllermeier, E.: ML-Plan for unlimited-length machine learning pipelines. In: *ICML 2018 AutoML Workshop*. 2018.

[WMTH19]   Wever, M. D.; Mohr, F.; Tornede, A.; Hüllermeier, E.: Automating multi-label classification extending ml-plan. In: *ICML 2019 AutoML Workshop*. 2019.

[WTMH21]   Wever, M.; Tornede, A.; Mohr, F.; Hüllermeier, E.: AutoML for Multi-Label Classification: Overview and Empirical Evaluation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), no. 9, pp. 3037–3054

[XHHL07]   Xu, L.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K.: SATzilla-07: The design and analysis of an algorithm portfolio for SAT. In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2007, pp. 712–727

Subproject B3:

Composition Analysis in Unknown Contexts

Matthias Becker[1], Steffen Becker[2], Eyke Hüllermeier[3], Cedric Richter[4],
Arnab Sharma[5], Heike Wehrheim[4]

1 Department of Computer Science, Fraunhofer IEM,
Paderborn, Germany

2 Department of Computer Science, University of
Stuttgart, Stuttgart, Germany

3 Institute of Informatics, LMU Munich, Munich,
Germany

4 Department of Computer Science, Oldenburg University,
Oldenburg, Germany

5 Department of Computer Science, Paderborn University,
Paderborn, Germany

Subproject B3 "Composition Analysis in Uncertain Contexts" deals with the quality assurance of service compositions as assembled by Subproject B2. Over the three funding periods, the objectives varied in the *type* of service composition, *type* of requirements and the *type* of analysis considered.

1 Introduction

Within the CRC, the task of Subproject B3 is to develop techniques for quality assurance of service compositions. Subproject B2 assembles service compositions from services traded on markets based on a requirements specification given by a user. For the interface between Subprojects B2 and B3, a common modeling language has been developed in the first period of the CRC. The task of B3 then consisted of analyzing the service composition before its execution.

In the first period of the CRC, the focus of the subproject has been on

- the development of a common modeling language for describing service compositions as used by Subprojects B1, B2 and B3,

- the analysis of non-functional properties (performance, scalability, and elasticity) of service compositions via simulations, and

- the analysis of functional properties as specified by pre- and postconditions for service compositions via logical encodings and SMT solving.

In the second period, Subproject B3 concentrated on

matthias.becker@iem.fraunhofer.de (Matthias Becker), steffen.becker@informatik.uni-stuttgart.de (Steffen Becker), eyke@lmu.de (Eyke Hüllermeier), cedric.richter@uni-oldenburg.de (Cedric Richter), arnab.sharma@uni-paderborn.de (Arnab Sharma), heike.wehrheim@uni-oldenburg.de (Heike Wehrheim)

- the use of *templates* for service compositions and analysis with the objective of speeding up quality assurance in an *on-the-fly* context,

- the localization of errors in service compositions when requirements are not met (again employing logical encodings and SMT solving), and

- the analysis of non-functional properties via machine learning techniques.

For the third period of the CRC, the focus shifted to considering service compositions with components generated by data-driven techniques. This was motivated by the type of compositions generated by Subproject B2 that started to concentrate on the generation of machine learning (ML) pipelines. Research in B3 then studied

- the analysis of data-driven systems with respect to specific (hyper-)properties,

- machine learning methods for the prediction of non-functional properties of service compositions that can be trained on-the-fly in an online (rather than batch) mode, as well as

- the increase of the robustness of such methods (e.g., against uncertainty or missing information about the context).

Throughout the entire funding period, all conceptual developments were complemented by tool implementations and extensive empirical evaluations. The research results have been published in international conferences and journals. In the following, we highlight some selected results of Subproject B3.

2 Main Contributions

2.1 Performance Prediction via Simulation

In on-the-fly computing scenarios, service compositions were assumed to happen at run-time and on demand. However, those compositions not only need to compose the right services horizontally (i.e., select a complete set of components which together fulfil all domain requirements) but also vertically. The latter means allocating the services on the right amount of resources, i.e., computing, storage and networking capacity.

Nevertheless, in contrast to classical static compositions, the environment of the service composition is unknown and can vary significantly over time. Hence, the allocation needs to adapt to the current environment based on quality requirements expressed via goals and formalized in service level objectives (SLOs).

In our research, we modeled not only these goals, SLOs, but also the composition and the self-adaptations, which always keep adjusting the allocation to the current environment. For these models we have developed a simulator that enables developers to judge the effectiveness of the composition and its self-adaptations upfront. We focused on performance and elasticity in our research and included adaptation strategies in our models and analyses that deal with elasticity.

As introduced above, performance and, more specifically, scalability and elasticity were the quality properties that we aimed to predict based on models of this service composition. For this purpose, a service composition model has to be enriched with performance-relevant

annotation. The contributions within Subproject B3 for this are described in the following
paragraphs.

Performance Modeling

In order to predict the performance of a service composition, we introduced a performance
modeling approach in [BBM13; BLB13] and further refined it with viewpoints and roles in
[Bec17]. With our performance modeling approach, we provide the necessary precondition
for the assessment of performance properties of service compositions: the extended
service composition model contains performance-relevant information, such as the resource
consumption of a service, as well as available resources of the service composition's
execution environment. Additionally, with service level objectives (SLOs), performance
demands for the execution of a service composition can be specified. A concrete workload
scenario and its evolution can be specified in order to simulate a service composition
execution and thus predict its scalability and elasticity.

Metrics for Scalability and Elasticity

We formally defined a service composition as a self-adaptive system that can be scalable
and elastic by adapting its architecture to its performance demands, specified as SLOs,
on-the-fly. The formalization is based on the Fuzzy Branching Temporal Logic [MLL04],
which allowed us to define a notion of graded SLO achievement, i.e., performance demands
of a service can be gradually fulfilled.

We defined scalability as the ability of a service composition to *eventually* adapt its capacity
to different workload scenarios without missing defined service level objectives. Elasticity
is the degree to which is service composition is able to self-adapt to workload scenarios,
such that it achieves all of its service level objectives to a certain *grade*. To quantify the
elasticity grade, we defined the two elasticity metric as *time to SLO achievment (TTSA)*
and *accumulated SLO achievement grade (ASAG)*.

TTSA is the metric that reflects the duration a service composition requires to achieve its
SLOs in a certain workload scenario. The duration is calculated as the difference from
the point in time when the service composition is in a specified state, e.g., its initial state,
until the point in time when the service composition is in a state in which its SLOs are
achieved. The base unit of the time to SLO achievement is defined as the base unit of time,
i.e., seconds (s).

ASAG is the metric that reflects the normalized, accumulated SLO achievement grade of a
service composition in a certain workload scenario. The ASAG value is calculated as the
(normalized) integral of the SLO achievement of a service composition over time from the
point in time when the service composition is in a specified state, e.g., its initial state, until
the point in time when the service composition is in a state in which its SLOs are achieved.
The metric has no unit, but the values are normalized and are in the interval between 0 and
1, i.e., interval [0; 1].

Prediction of Scalability and Elasticity

Based on our formalization and on our metric definitions, we provided prediction methods
for our scalability and elasticity metrics based on a performance simulation of the service

composition. Figure 27 illustrates the states of a scalable service composition. The starting state of the simulation Σ_0 is given by the service composition model, see step (1). The state transitions α_i are also defined in the model as model transformations. In order to predict *scalability* of the service composition, each state that is reachable via a model transformation is simulated, see steps (2) and (3) in Figure 27. In each simulation it is checked whether all defined SLOs are achieved eventually, i.e., in a stable performance state. This is repeated until a state is reached that fulfills all SLOs or no more states can be explored, see step (4) in Figure 27. The *elasticity* is predicted by starting a performance simulation in the initial state Σ_0 and applying model transformations during this simulation whenever a precondition of a state transition is met. In this way, the elasticity metrics described above can be determined.

(1) *Check whether SLOs are achieved in the current state.*

(2) *Execute an architecture reconfiguration and check if SLOs are achieved.*

(3) *Execute next architecture reconfiguration and check whether SLOs are achieved.*

(4) *Continue with next unexplored state at step (1).*

Legend
○ unexplored state   ● analyzed, unexplored state   ● analyzed, explored state
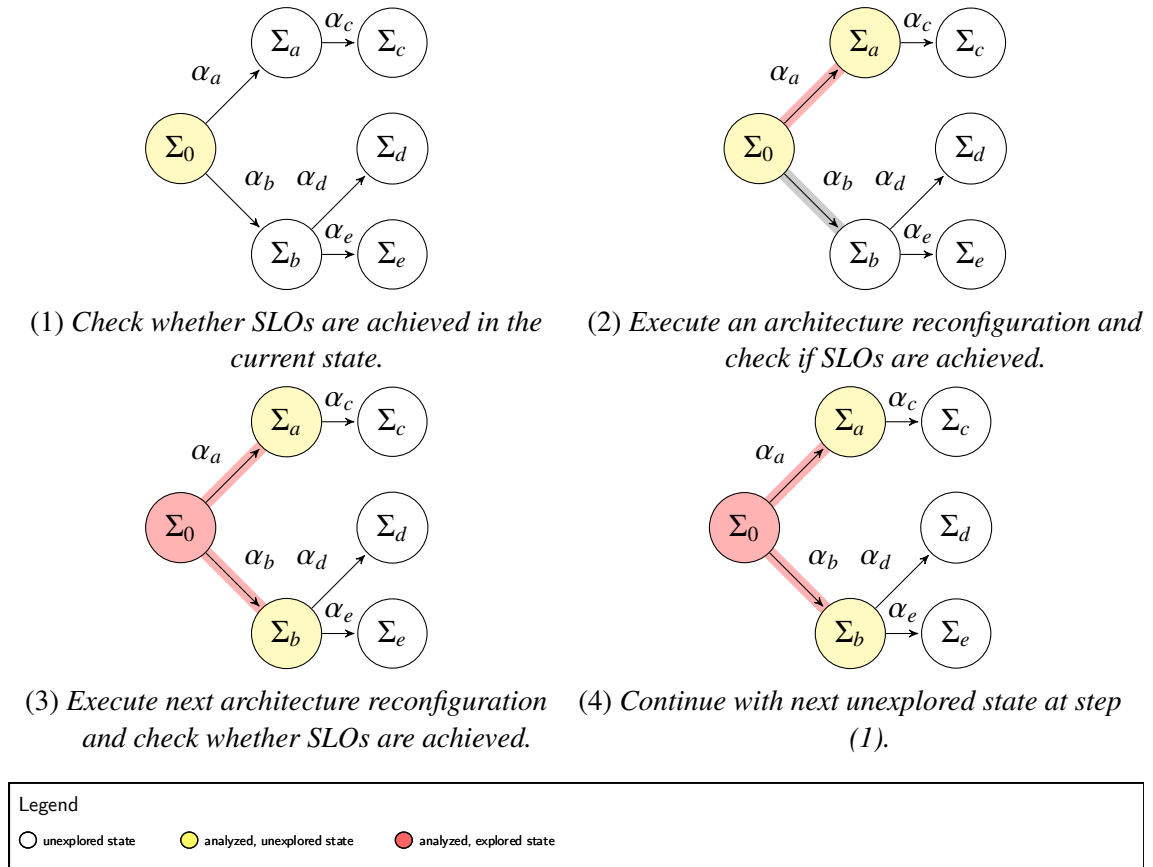
Figure 27: *Exploration of scalability.*

2.2 Algorithm Selection for Software Verification

During the second phase of the CRC, Subproject B3 investigated machine learning techniques for *selecting* analysis techniques. More specifically, we looked at various techniques for software verification and studied the question of *algorithm selection*, i.e., how to select an appropriate technique for a verification task at hand [CHJW17; RHJW20; RW19]. Even though software verification is a mature field and a lot of software verification algorithms
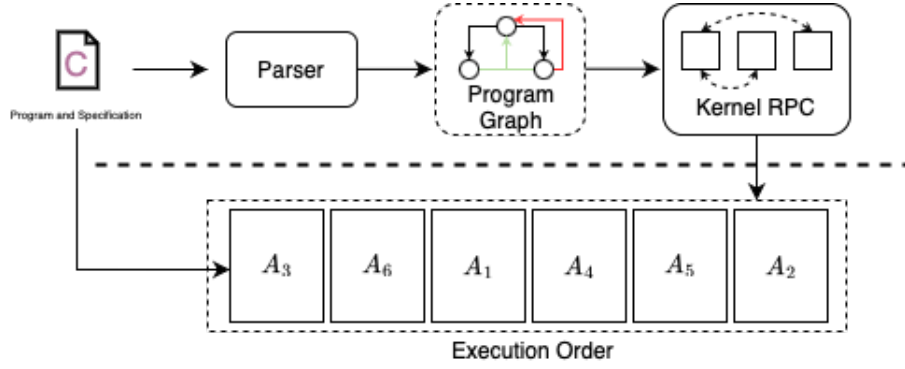
Figure 28: *Overview of the PeSCo framework.*

have been developed over the past decades [BDW15; BF16; BKW10; HCD$^+$13; CJS$^+$16], this is an important question as there is no single algorithm that dominates all other verification algorithms on all possible verification problems. Therefore, we (often manually) have to pick the right algorithm for a given verification task.

To automate the selection process, we developed an approach for predicting the *task-specific* performance of software verification algorithms [RHJW20]. An accurate prediction can then help us to automatically identify and select the best performing algorithm for a given task. In the following, we describe the approach and its instantiations in more detail.

Learning to Select Verifiers

We assume that a verification task consists of a program P and a specification φ. The software verification algorithm, or software verifier for short, then has to verify whether the program satisfies the specification or not. Note that in reality the verification process is limited by system resources and the verifier can only be successful if it verifies a given task within a certain amount of time or memory.

Now, given a set of verifiers $\mathbb{A} = \{A_1, A_2, \ldots, A_n\}$, our goal is to identify the verifier that verifies the given verification task within the given resource constraints. For this, we employed a machine learning model that learns to "guess" the performance of the individual algorithms and then rank them accordingly. We then select the highest ranked verification algorithm.

However, to design such a learning based model that can predict the performance of verifiers, we had to overcome two key challenges:

1. How to represent programs and specifications such that we can infer the performance of verifiers?

2. How to integrate our representation into classical machine learning pipelines?

In contrast to previous work [TKK$^+$14; DPVZ17], we decided against representing the verification tasks as feature vectors directly and choose a representation that is closer to the internal representations used inside verifiers. In fact, our approach transforms a given verification task into a combination of an abstract syntax tree, control flow graph and

program dependency graph [HR92]. In our case, the specification is encoded inside the program and therefore indirectly represented through the graph structure.

To integrate our program representation into the learning process, we employed a kernel based method [SS02] that enabled us to directly learn on graph representations without an extra feature extraction process. In other words, by employing kernel-based methods, our model learns which graph structures are important for predicting the performance of verifiers. For this, we introduced a custom kernel and utilized kernelized support vector machines [SS02] for the learning process.

During training, our learner learns to rank verification algorithms via the ranking by a pairwise comparison (RPC) framework [FH10]. Here, the learning task is decomposed into multiple binary classification problems. Each resulting classifier then predicts whether a verifier A_i performs better on the given task than another verifier A_j. We define that a verifier A_i is better than a verifier A_j on a given task (and therefore ranked higher) if A_i is more likely to solve the task within the given resource constraints or both verifiers solve the task equally likely but A_i is likely faster.

Finally, we employ the learned model to predict the most likely best performing verifier for a given task. An overview of the prediction process is shown in the upper part of Figure 28. For a new verification task, we first parse the given program and specification into the graph representation. The graph representation is then provided to the learned Kernel RPC model which predicts a ranking of verifiers.

Predicting Sequential Compositions of Verifiers

We implemented our selection approach inside the verification tool CPAchecker [BK11], which ultimately resulted in a new verification tool called PeSCo [RW19]. PeSCo ranks up to six base verification algorithms and then executes them in order. As a result, PeSCo is able to select from over 15 different sequential verifier compositions based on the characteristics of the given verification task.

In addition, we found that performance modeling for ranking verification algorithms is also effective in practice. With its selection approach, PeSCo won the second place in the overall category of the 8th international software verification competition (SVComp) [Bey19] and since then remains highly successful in the competition.

As part of a DFG-funded project on "Cooperative Verification" we continue the work of algorithm selection for software verification, now with a focus on selecting components for a cooperative approach.

2.3   Functional Analysis of Service Compositions

In the first two phases of Subproject B3, we considered the analysis of service compositions specified in the common modeling language, jointly developed between Subprojects B1, B2 and B3 [AWBP14]. The focus was on the analysis of functional properties specified via pre- and postconditions for service compositions. The compositions are assembled out of single services traded on the market. Each such service has a specification written
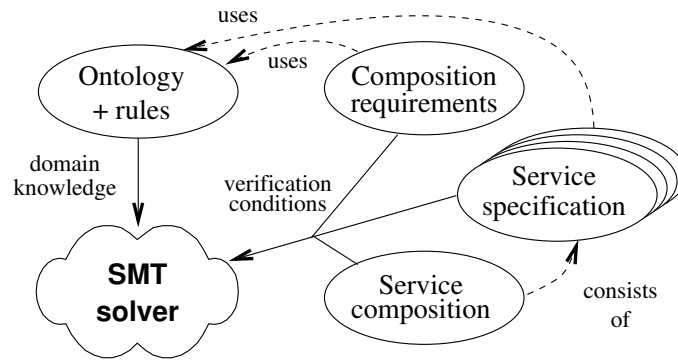
Figure 29: *Overview of the ontology-based approach*

in the modeling language as well. The vocabulary of the modeling language is based on a domain-specific ontology. This accounts to types used in service signatures, but also to predicates occurring in preconditions and effects of services. Ontologies, in particular those enhanced with *rules*, capture the knowledge of domain experts on properties of and relations between domain concepts.

Our verification technique for service compositions [WW13] makes use of this domain knowledge. We consider a service composition to be an assembly of services of which we just know signatures, preconditions, and effects. Compositions are written in a simple workflow language, such as specifiable via activity diagrams. We aim at proving that a composition satisfies a (user-defined) requirement, specified in terms of guaranteed preconditions and required postconditions. For an underlying verification engine we use an SMT solver. More specifically, we translate single service specifications, the service composition and the ontology rules to first order predicate logic to be fed into an SMT solver (see Figure 29). Similarly, we translate the user requirement into a logical formula. To take advantage of the domain knowledge (and often, to *enable* verification at all), the knowledge is fed into the solver in the form of sorts, uninterpreted functions and, in particular, assertions as to enhance the solver's reasoning capabilities. Thereby, we allow for deductions within a domain previously unknown to the solver. In the CRC, we have applied our technique on a case study from the area of water network optimization software (as studied by Subproject C3 on "Modeling of Optimization Problems" in the first phase). In the following, we describe the technique in more detail.

Verification Approach

We assume a given composition of services, each with an ontology-based interface specification. Apart from interfaces, nothing is known about the services (black-box view). In the context of service-oriented architectures (SOA), this is a quite likely scenario: Providers sell their services but not the code itself. In fact, a service might not even run on the consumer side, but could either completely stay on the provider machine or run in the cloud. Furthermore, the requirements on an assembled service composition are specific to the domain; instead of proving general safety or reachability properties alone (as state-of-the-art software verification tools do), consumers expect the verification to prove domain-specific requirements. We leverage this by grounding service specifications on ontologies.
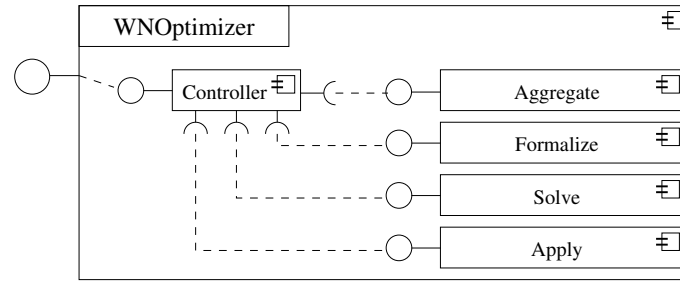
Figure 30: *Service composition of the WaterNet Optimizer.*

We exemplify this with the case study "water supply network optimization" (see Figure 30 for a service composition in this domain). Software services in this domain handle different tasks of analyzing and optimizing existing municipal water supply networks. Single services designed for different subtasks can be assembled into a composition. This concerns services such as (a) compacting the size (and layout) of network models, (b) generating mathematical optimization problems from networks, (c) solving optimization problems, and (d) applying optimal solutions to networks. As the behavior of these services is specified in terms of interfaces only, this is a black-box view and for the analysis we can therefore only assume that services adhere to their specification.

Every water network has specific hydraulic characteristics, as well as other properties such as the cost of operation. A typical domain specific requirement on a composition of some optimization services is that an optimized network (produced by the composition) has the *same* hydraulic characteristics as the original input network, but a *better* (e.g., lower) cost of operation. The verification technique has to derive this property from the given service specifications, the way of assembling the services and the additional domain knowledge stored in the ontology.

Our approach to the verification of such a service composition is based on the use of an SMT solver (satisfiability modulo theories solver) as reasoning engine. Basically, our technique feeds three types of inputs (domain knowledge, service interface specifications, and assembly) into the SMT solver in different forms (see Figure 29). These inputs, combined with the user's requirements specification, are encoded as first-order logic formulae. In this encoding, the user requirement is negated. The resulting formula is then checked for satisfiability: If unsatisfiable, the requirement is fulfilled; if satisfiable, a counterexample is found.

More specifically, we start with an ontology of the domain which – besides the standard concepts and their relations – models additional rules about the domain by first-order logic. The predicates therein are the relations in the ontology. Providers of services use the ontology to specify a service's signature and its preconditions and effects. Consumers use the ontology to specify requirements of a service composition. For verification, we use the concepts of the ontology as *types* for the solver, relations as *uninterpreted functions*, and rules as constraints on the interpretation of these functions. The rules are thus being used for deduction together with the decidable theories of the solver (e.g., linear arithmetic). The creation of verification conditions for a given service composition and requirements follows ideas of Hoare-style proofs. It turns out that the verification typically requires the additional domain knowledge for a successful reasoning: The knowledge of human domain experts (e.g. about hydraulic properties of different forms of networks) needs to be

provided to the solver.

Templates

In cooperation with Subproject B2 [MW15], the basic verification approach was complemented with the idea of *templates* [WW14; WW16]. This was motivated by the on-the-fly principle, because pre-verified templates upon instantiation only require checks of the soundness of the instantiation, not of the entire composition. Templates can capture known composition patterns, and thus allow for the application of the general principle of pattern usage in software engineering.

More specifically, following our approach for the verification of service compositions, templates are workflow descriptions with *service placeholders*. Service placeholders are replaced by concrete services during instantiation. If a template is shown to be correct, then all of its (valid) instantiations will be *correct by construction*. Every template specification contains functional properties given in terms of pre- and postconditions (again with associated meaning "if precondition fulfilled, then postcondition guaranteed"), and a correct template provably adheres to this specification. To verify correctness of templates, we employ the Hoare-style proof calculus as of above.

The definition of "correctness" as well as giving a proof calculus for templates, however, poses a non-trivial task on verification. Since templates should be usable in a wide range of contexts and the instantiations of service placeholders are unknown at template design time, we cannot give a fixed semantics to templates. Rather, the template semantics needs to be *parameterized* in usage context and service instantiation. A template is only correct if it is correct for all (allowed) usage contexts. Similarly, a useful proof calculus has to be applicable in all possible contexts and service instantiations. We guarantee this by defining a proof calculus that is *parameterized* in usage contexts and template-specific constraints.

Technically, we capture the usage contexts by ontologies, and the interpretation of concepts and predicates occurring therein by logical structures. A *template ontology* defines the concepts and predicates of a template. Furthermore, a template specification contains constraints defining additional conditions on instantiations. These constraints allow us to verify the correctness of the template despite unknown usage and unknown fixed semantics. A template instantiation replaces the template ontology with a homomorphous domain ontology, and the service placeholders with concrete services of this domain. Verification of the instantiation then amounts to checking whether the (instantiated) template constraints are valid within the domain ontology, and thus can be carried out on-the-fly.

2.4 Performance Prediction via Machine Learning

As an alternative to the use of simulation techniques (cf. Section 2.1), the potential of machine learning (ML) methods for non-functional analysis and performance prediction has been investigated in the second and third funding period. The idea here is to induce models that predict a property of a service composition, given the specification of the service as input. What makes this problem challenging from an ML perspective is the

specific structure of service compositions: Services are recursively structured objects of variable size. Representing them in terms of feature vectors of fixed length, the format commonly assumed by most ML methods, is difficult and will necessarily cause a loss of information.

To cope with these challenges, we introduced a new ML setting that we call "learning to aggregate" (LTA). Roughly, learning-to-aggregate problems are supervised machine learning problems in which data objects are represented in the form of a *composition* of a (variable) number on *constituents*; such compositions are associated with an evaluation, score, or label, which is the target of the prediction task, and which can presumably be modeled in the form of a suitable aggregation of the properties of its constituents. Thus, our LTA framework establishes a close connection between machine learning and a branch of mathematics devoted to the systematic study of aggregation functions [GMMP09].

A bit more formally, we proceed from a set of training data $\mathcal{D} = \{(c_1, y_1), \ldots, (c_N, y_N)\} \subset C \times \mathcal{Y}$, where C is the space of compositions and $\mathcal{Y}$ a set of possible (output) values associated with a composition. Since aggregation is often used for the purpose of evaluating a composition, we also refer to the values y_i as *scores*. A composition $c_i \in C$ is a multiset (*bag*) of constituents $c_i = \{c_{i,1}, \ldots, c_{i,n_i}\}$, where $n_i = |c_i|$ is the size of the composition; scores y_i are typically scalar values (e.g., representing a specific non-functional property of a service). Constituents $c_{i,j}$ can be of different type, and the description of a constituent may or may not contain the following information:

- A *label* specifying the role of the constituent in the composition. For example, suppose a composition is a service in the form a machine learning pipeline (cf. Subproject B2) consisting of an algorithm for data preprocessing, a method for inducing a classifier, and an algorithm for postprocessing predictions. By assigning labels to these constituents, such as `pre`, `induce`, and `post`, additional information is provided about the part of the composition they belongs to (thereby adding additional structure to the composition).

- A description of *properties* of the constituent, for example, memory requirements of an algorithm. Formally, we assume properties to be given in the form of a feature vector $v_{i,j} \in \mathcal{V}$, where $\mathcal{V}$ is a corresponding feature space. However, more complex descriptions are conceivable. For example, the description could itself be a composition.

- A *local evaluation* in the form of a score $y_{i,j} \in \mathbb{R}_+$.

Finally, a composition can also be equipped with an additional structure in the form of a (binary) relation on its constituents. In this case, a composition is not simply an unordered set (or bag) of constituents but a more structured object, such as a sequence (like in the above example of an ML pipeline) or a graph.

Like in standard supervised learning, the goal in learning-to-aggregate is to induce a model $h : C \longrightarrow \mathcal{Y}$ that predicts scores for compositions. More specifically, given a hypothesis space $\mathcal{H}$ and a loss function $L : \mathcal{Y}^2 \longrightarrow \mathbb{R}_+$, the goal is to find a hypothesis $h^* \in \mathcal{H}$ that provides optimal predictions in the sense of minimal L in expectation.
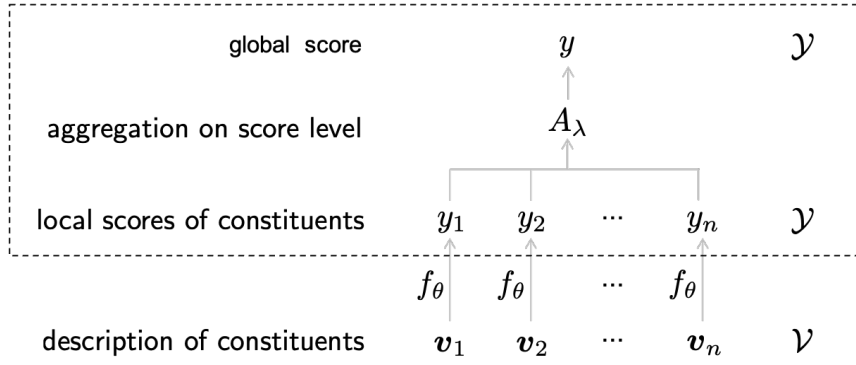
Figure 31: *Illustration of a basic version of the learning-to-aggregate framework.*

Learning Aggregation Functions

One of the key problems in learning to aggregate is to combine a variable number of scores $y_{i,j}$, pertaining to evaluations of the constituents $c_{i,j}$ in a composition $\boldsymbol{c}$, into a single score y_i. In Figure 31, which provides an overview of the LTA setting, this step corresponds to the part marked by the dashed rectangle.

Now, suppose that we know or can at least reasonably assume that y_i is obtained from $y_{i,1}, \ldots, y_{i,n_i}$ through an aggregation process defined by a binary aggregation function $A : \mathcal{Y}^2 \longrightarrow \mathcal{Y}$:

$$y_i = A\Big(\ldots A(A(y_{i,1}, y_{i,2}), y_{i,3}), \ldots, y_{i,n_i} \Big).$$

In the simplest case, where the constituents do not have labels and hence cannot be distinguished, the aggregation should be invariant against permutation of the constituents in the bag. Thus, it is reasonable to assume A to be associative and symmetric. Besides, one may of course restrict an underlying class of candidate functions $\mathcal{A}$ by additional assumptions, such as monotonicity.

Starting from a class $\mathcal{A}$ of aggregation functions, instead from a hypothesis space $\mathcal{H}$ directly, has at least two important advantages. First, as just said, it allows for incorporating prior knowledge about the aggregation, which may serve as a suitable inductive bias of the learning process. Second, it naturally solves the problem that hypotheses $h \in \mathcal{H}$ must accept inputs of any size. Indeed, under the assumption of associativity and symmetry, a binary aggregation function A is naturally extended to any arity, and can hence be used as a "generator" of a hypothesis $h = h_A$:

$$h(y_1, \ldots, y_n) = A^{(n)}(y_1, \ldots, y_n) = A\Big(A^{(n-1)}(y_1, \ldots, y_{n-1}), y_n \Big)$$

for all $n \geq 1$, where $h(y_1) = A^{(1)}(y_1) = y_1$ by definition. For these reasons, we consider the learning of (binary) aggregation functions, and related to this the specification of a suitable class $\mathcal{A}$ of candidates, as an integral part of learning to aggregate.

Disaggregation

The aggregation we have been speaking about so far is an aggregation on the level of scores. Thus, we actually assume that local scores $y_{i,j}$ of the constituents $c_{i,j}$ are already given

and that we are interested in aggregating them into an overall score y_i of the composition c_i. This is indeed the genuine purpose of aggregation functions, which typically assume that all scores are elements of the same scale $\mathcal{Y}$. Now, suppose that local scores $y_{i,j}$ are not part of the training data. Instead, the constituents $c_{i,j}$ are only described in terms of properties in the form of feature vectors $v_{i,j} \in \mathcal{V}$. A natural way to tackle the learning problem, then, is to consider the local scores as latent variables, and to induce them as functions $f : \mathcal{V} \longrightarrow \mathcal{Y}$ of the properties.

More specifically, we assume these functions to be parameterized by a parameter vector θ, and the aggregation function A by a parameter λ. The model is then of the form

$$y_i = A_\lambda(y_{i,1}, \ldots, y_{i,n_i}) = A_\lambda\Big(f_\theta(v_{i,1}), \ldots, f_\theta(v_{i,n_i})\Big) \ ,$$

and the problem consists of learning both the aggregation function A, i.e., the parameter λ, and the mapping from features to local scores, i.e., the parameter θ, simultaneously. Here, supervision only takes place on the level of the entire composition, namely in the form of scores y_i, whereas the "explanation" of these scores via induction of local scores is part of the learning problem.

The decomposition of global scores into several local scores is sometimes referred to as *disaggregation* (because it inverts the direction of aggregation, which is from local scores to global ones). One could then try to learn how the constituents are rated (via f_θ) and, simultaneously, how the corresponding local scores are aggregated into a global rating (via A_λ). Obviously, there is a strong interaction between local rating and aggregation on a global level. An important question, therefore, concerns the *identifiability* of the model, i.e., the question whether different parameterizations imply different models (or, more formally, whether $(\lambda, \theta) \neq (\lambda', \theta')$ implies that the corresponding models assign different scores $y_i \neq y_i'$ for at least one composition).

Instantiations

The LTA framework as outlined above has been instantiated in different ways and evaluated on practical learning tasks. A first instantiation based on a class of aggregation functions called *uninorms* has been proposed in [MH16]. Learning algorithms for another type of aggregation function, so-called *ordered weighted averaging operators*, have been developed and tested in [MH19].

2.5   Testing of Data-Driven Software Systems

In the third phase of the CRC, the service compositions to be analyzed by Subproject B3 were *pipelines* of machine learning components, such as data generation, preprocessing, learning etc. generated by Subproject B2. Essentially, through the pipeline of such services, B2 generates data-driven software systems (DSS).

Unlike traditional software systems, where intended behavior of the software is programmed by the developer, data-driven software *learns* its intended functionality from lots of examples. The analysis of such a system faces two fundamental challenges: (1) identification of the requirements to be checked and (2) development of an analysis method.

The first challenge arises out of the fact that the actual intended behavior of the learned component is unclear as otherwise learning would not be required at all. The second challenge arises because learning algorithms generate a diverse set of different classifiers (or regressors).

More precisely, given a set of data instances (also called a *training data set*), a machine learning algorithm generalizes from the data set thereby generating a machine learning (ML) *model* (or DSS[11]). Formally, this model is a mapping from inputs to an output, i.e.,

$$M : X_1 \times \ldots \times X_n \to Y \, .$$

The X_i denotes the value set of the input element (also called the *feature*) i and Y denotes the set of output values. However, it is essentially unclear what the correct outcome of this process is, i.e., what is considered to be an expected model M. Moreover, even if we can identify some requirements to check, there can be different types of ML models as the outcome of the learning process, depending on the learning algorithm used, such as decision tree, neural network, random forest, support vector machine or others.

In recent years, with the increased usage of such data-driven software, there have been a number of works focusing on ensuring the quality of such data-driven systems (see e.g., [ZHML22; Alb21]). To this end, two approaches are currently followed: a) developing an ML algorithm guaranteeing a requirement per design or b) validating the requirement on a given DSS. There are shortcomings for both of these approaches. Firstly, the requirement-per-design algorithms are only available for a small number of requirements. Moreover, it has been found out that in some cases these algorithms were unable to guarantee the desired requirements [GBM17]). Secondly, validation techniques are either restricted to a specific model or to a specific requirement to check, such as checking fairness for deep neural network model [ZWS+20].

Within Subproject B3, we have proposed a validation technique called *property-driven testing* with the intention of overcoming the shortcomings of existing techniques. Our method is a *validation* technique in that we aim at the falsification of requirements, i.e., finding counterexamples to properties like standard testing techniques do. Contrary to standard testing often using random generation of test inputs, we however have a systematic, verification-based technique for generating potential counterexamples. Our technique is "property-based" as it allows the checking of user-supplied properties, written in a pre- and postcondition format. We have implemented this testing approach in a tool named MLCHECK and have evaluated it to check a number of properties on several types of ML models. All the code and data of this work is publicly available at https://github.com/arnabsharma91/MlCheck. Next, we briefly describe the steps involved in our property-driven testing framework.

Property-Driven Testing

We have developed a testing mechanism that allows the user to specify the property using a standard specification language that would then be used for test case generation. To this end, we have the following two contributions: a) a domain-specific property specification language and b) a targeted test case generation method.

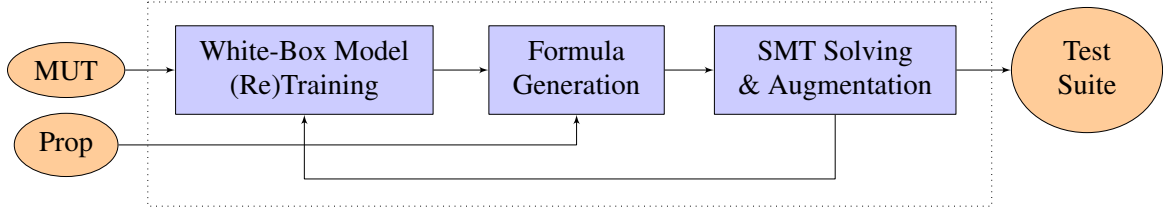[11]We use the term ML model and data-driven software (DSS) interchangeably in this section.

Figure 32: *Workflow of test data generation*

Property specification. We give an *assume/assert* style specification language where an assume statement specifies a condition on the input and assert statement specifies the condition to be satisfied by the output of an ML model. We develop this considering Python as a base programming language because of its high use in developing data-driven software. Essentially, assume/assert statements are defined as calls to the functions `Assume` and `Assert` respectively and take the following form [SDNW21].

```
Assume('<condition>',<arg1>, ...)
Assert('<condition>',<arg1>, ...)
```

The first parameter is a string defining a logical condition on the input data instance (for `Assume`) or the output (for `Assert`) of the model under test (MUT), combining any other variables in the code. The rest of the arguments give the values of these variables respecting the order of their occurrences in the condition. Later, this condition, along with the values of the variables, is translated to a logical formulae. Further details about our specification language and its grammar can be found here [SMHW22].

Test case generation. We perform this step employing a technique called *verification-based* testing, which we propose in [SW20]. To this end, first of all, we generate a set of data instances randomly and for each of these instances we get the corresponding predictions (i.e., outputs) from the MUT. The set of instances, along with their predictions form the training data set for a *white-box model* (see Figure 32). The newly generated white-box model in our framework can be either a decision tree or a neural network model. It approximates the MUT. Next, we convert white-box model W and the negation of the property specification to a logical formula (in SMT-LIB format[12]) ϕ_W and $\phi_{\neg P}$ respectively and we conjunct them to get $\phi_W \land \phi_{\neg P}$. This translation to logical formula guarantees to give a satisfiable formula if and only if the white-box model does not satisfy the property. The conjuncted formula then is given to the satisfiability modulo theory (SMT) solver Z3 [MB08]. If the Z3 finds the formula to be satisfiable, it will return a counterexample to the property, i.e., an input to W that shows the violation of the given property.

Now, this counterexample serves as a test case and, using a method called *pruning*, we generate more of these. However, as we find the counterexamples on W, not on the MUT, we must check the validity of the counterexamples on the MUT. In case they are not valid, we add the input instances from the counterexamples, along with its real predictions from MUT to the training data set and retrain the model W to get a better approximation of the MUT. Otherwise, we store the counterexamples and return them as counterexamples for the MUT M. These steps are repeated until a user defined timeout occurs.

---

[12]http://smtlib.cs.uiowa.edu/

Results

We have implemented the property-driven testing approach in a tool called MLCHECK and applied it to check for several types of properties. For example, in [SW20], we applied our approach to test monotonicity requirements of ML models. Our evaluation shows that our approach outperforms adaptive random testing [CLM04] and property-based testing [CH00] approaches in finding out monotonicity violations. Furthermore, our approach can find out the violations even for ML models that are by designed to be monotonic. We also checked for several types of fairness criteria in [SW20] and found our approach to be effective in finding out more number of fairness violations than the existing fairness testing approaches [ALN+19; UAC18]. Our approach shows that existing learning algorithms that are by design meant to be fair can generate unfair models, leading to fairness violation. In a later work, we furthermore checked security and concept relationship requirements (developed in cooperation with Subproject B2) of data-driven software [SDNW21]. Finally, in a recent work we used MLCHECK to evaluate a number of mathematical properties on a specific type of ML models (i.e., *regression* models) [SMHW22]. In this case, the requirements reflect properties of aggregation functions as studied within B3 in the context of "learning to aggregate" [MH16; MH19]. Thus, we can apply our tool in testing diverse properties for several types of data-driven software systems.

3   Impact and Outlook

The research conducted in this Subproject over the last decade, and notably the contributions highlighted in this chapter, has been impactful and has triggered follow-up work by ourselves and other scholars.

For example, based on our research on the prediction of scalability and elasticity (which ended after the first period due to the leave of PI Becker), several follow-up projects pushed these ideas further. The EU FP7 project CloudScale extended the presented simulation approach to analyze SLO achievement by architectural templates (ATs), which makes it much easier for end users to model typical elasticity patterns in cloud computing allocations. Becker and his colleagues also contributed a pattern catalogue containing patterns for horizontal and vertical scale-up/-down and scale-out/-in including the corresponding load balancing strategies.

When using the approach in practice, it was realized that it can be rather difficult to analyze the simulator's results and to improve the self-adaptation rules based on these results alone. Hence, in a current ongoing DFG project, we aim at explainability of the simulator's results. The vision is that, based on the simulator's results, the system should explain which self-adaptations have been taken when and why. Ideally, it might even make suggestions on how to change the self-adaptation rules to achieve improved results.

Another example of impactful research is our work on algorithm selection for software verification. In particular, the development of the tool PeSCo has inspired the development of other algorithm selectors. We ourselves have shown that approaches based on neural networks can be used to learn *transferable* feature representation, applicable to many verifier selection scenarios [RW20]. Apart from us, Beyer et al. [BKR22] have found

that *combinations* of complete verification tools chosen via algorithm selection significantly outperform the performance of single tools. Finally, a new verification tool called GraVeS [LD22] has been developed based on the PeSCo architecture, and has already been evaluated successfully in the software verification competition [Bey22].

Our work on machine learning for predictive modeling of service properties has triggered follow-up work, too. In particular, the "learning-to-aggregate" setting that we introduced has inspired other researchers. Obviously, this setting is not restricted to the prediction of properties of service compositions, but can also be applied to other learning tasks, where global scores are naturally modeled as an aggregation of local evaluations. In [PTF+21], for example, the LTA framework has been picked up and extended by the introduction of so-called learnable aggregation functions (LAF) for sets of any cardinality. This class of functions is shown to be very versatile and able to approximate many important aggregators in a flexible way. In experimental studies, the approach has been compared to other methods for learning from sets, and was found to outperform state-of-the-art approaches from the field of deep neural networks.

On a broader scale, the importance of the research topics addressed in this subproject is even likely to increase in the near future, especially due to the rapid development in the field of artificial intelligence. With the quick expansion of practical AI applications, along with the increasing trend toward the data-driven construction of AI tools based on neural network technology, the verification of these tools is becoming more and more crucial. We initialized work in this direction in the third funding period, but of course, this can mark just the start of a bigger research program. Currently, for example, there is a lot of work on formal verification of neural networks, motivated by the need to provide formal guarantees on the correctness, safety, robustness, or fairness of such networks. In a sense, verification goes hand in hand with other approaches aimed at increasing the trustworthiness of AI systems, such as explainability. We believe that the methods and tools developed in this subproject provide a suitable basis for further developments in this field.

Bibliography

[Alb21]   ALBARGHOUTHI, A.: Introduction to Neural Network Verification. In: *Found. Trends Program. Lang.* 7 (2021), no. 1-2, pp. 1–157.

[ALN+19]   AGGARWAL, A.; LOHIA, P.; NAGAR, S.; DEY, K.; SAHA, D.: Black box fairness testing of machine learning models. In: *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE.* Ed. by DUMAS, M.; PFAHL, D.; APEL, S.; RUSSO, A. ACM, 2019, pp. 625–635.

[AWBP14]   ARIFULINA, S.; WALTHER, S.; BECKER, M.; PLATENIUS, M. C.: SeSAME: modeling and analyzing high-quality service compositions. In: *ASE.* Ed. by CRNKOVIC, I.; CHECHIK, M.; GRÜNBACHER, P. ACM, 2014, pp. 839–842.

[BBM13]   BECKER, M.; BECKER, S.; MEYER, J.: SimuLizar: Design-Time modeling and Performance Analysis of Self-Adaptive Systems. In: *Proceedings of the Software Engineering Conference (SE).* Lecture Notes in Informatics (LNI). 2013, pp. 71–84

[BDW15]   BEYER, D.; DANGL, M.; WENDLER, P.: Boosting k-Induction with Continuously-Refined Invariants. In: *Computer Aided Verification - 27th International Conference, CAV.* Ed. by KROENING, D.; PASAREANU, C. S. Vol. 9206. Lecture Notes in Computer Science. Springer, 2015, pp. 622–640.

[Bec17]   BECKER, M.: Engineering Self-Adaptive Systems with Simulation-Based Performance Prediction. PhD thesis. Universität Paderborn, Softwaretechnik, 2017

[Bey19]   BEYER, D.: Automatic Verification of C and Java Programs: SV-COMP 2019. In: *Tools and Algorithms for the Construction and Analysis of Systems TACAS TOOLympics, Held as Part of ETAPS Part III*. Ed. by BEYER, D.; HUISMAN, M.; KORDON, F.; STEFFEN, B. Vol. 11429. Lecture Notes in Computer Science. Springer, 2019, pp. 133–155.

[Bey22]   BEYER, D.: Progress on Software Verification: SV-COMP 2022. In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part II*. Ed. by FISMAN, D.; ROSU, G. Vol. 13244. Lecture Notes in Computer Science. Springer, 2022, pp. 375–402.

[BF16]   BEYER, D.; FRIEDBERGER, K.: A Light-Weight Approach for Verifying Multi-Threaded Programs with CPAchecker. In: *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS*. Ed. by BOUDA, J.; HOLÍK, L.; KOFRON, J.; STREJCEK, J.; RAMBOUSEK, A. Vol. 233. EPTCS. 2016, pp. 61–71.

[BK11]   BEYER, D.; KEREMOGLU, M. E.: CPAchecker: A Tool for Configurable Software Verification. In: *CAV*. Ed. by GOPALAKRISHNAN, G.; QADEER, S. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 184–190.

[BKR22]   BEYER, D.; KANAV, S.; RICHTER, C.: Construction of Verifier Combinations Based on Off-the-Shelf Verifiers. In: *Fundamental Approaches to Software Engineering -FASE 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*. Ed. by JOHNSEN, E. B.; WIMMER, M. Vol. 13241. Lecture Notes in Computer Science. Springer, 2022, pp. 49–70.

[BKW10]   BEYER, D.; KEREMOGLU, M. E.; WENDLER, P.: Predicate abstraction with adjustable-block encoding. In: *Proceedings of 10th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2010, Lugano, Switzerland, October 20-23*. Ed. by BLOEM, R.; SHARYGINA, N. IEEE, 2010, pp. 189–197.

[BLB13]   BECKER, M.; LUCKEY, M.; BECKER, S.: Performance Analysis of Self-Adaptive Systems for Requirements Validation at Design-Time. In: *Proceedings of the 9th ACM SigSoft International Conference on Quality of Software Architectures (QoSA'13)*. 2013, pp. 43–52

[CH00]   CLAESSEN, K.; HUGHES, J.: QuickCheck: a lightweight tool for random testing of Haskell programs. In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00)*. Ed. by ODERSKY, M.; WADLER, P. ACM, 2000, pp. 268–279.

[CHJW17]   CZECH, M.; HÜLLERMEIER, E.; JAKOBS, M.; WEHRHEIM, H.: Predicting rankings of software verification tools. In: *Proceedings of the 3rd ACM SIGSOFT International Workshop on Software Analytics, SWAN@ESEC/SIGSOFT FSE*. Ed. by BAYSAL, O.; MENZIES, T. ACM, 2017, pp. 23–26.

[CJS⁺16]   CHALUPA, M.; JONÁS, M.; SLABY, J.; STREJCEK, J.; VITOVSKÁ, M.: Symbiotic 3: New Slicer and Error-Witness Generation - (Competition Contribution). In: *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS nProceedings*. Ed. by CHECHIK, M.; RASKIN, J. Vol. 9636. Lecture Notes in Computer Science. Springer, 2016, pp. 946–949.

[CLM04]   CHEN, T. Y.; LEUNG, H.; MAK, I. K.: Adaptive Random Testing. In: *Advances in Computer Science - ASIAN 2004, Higher-Level Decision Making, 9th Asian Computing Science Conference, Proceedings*. Ed. by MAHER, M. J. Vol. 3321. Lecture Notes in Computer Science. Springer, 2004, pp. 320–329.

[DPVZ17]   DEMYANOVA, Y.; PANI, T.; VEITH, H.; ZULEGER, F.: Empirical software metrics for benchmarking of verification tools. In: *Formal Methods Syst. Des.* 50 (2017), no. 2-3, pp. 289–316.

[FH10]　　　　Fürnkranz, J.; Hüllermeier, E.: Preference Learning and Ranking by Pairwise Comparison. In: *Preference Learning*. Ed. by Fürnkranz, J.; Hüllermeier, E. Springer, 2010, pp. 65–82.

[GBM17]　　　Galhotra, S.; Brun, Y.; Meliou, A.: Fairness testing: testing software for discrimination. In: *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE*. Ed. by Bodden, E.; Schäfer, W.; Deursen, A. van; Zisman, A. ACM, 2017, pp. 498–510.

[GMMP09]　　Grabisch, M.; Marichal, J.; Mesiar, R.; Pap, E.: *Aggregation Functions*. Cambridge University Press, 2009

[HCD+13]　　Heizmann, M.; Christ, J.; Dietsch, D.; Ermis, E.; Hoenicke, J.; Lindenmann, M.; Nutz, A.; Schilling, C.; Podelski, A.: Ultimate Automizer with SMTInterpol - (Competition Contribution). In: *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*. Ed. by Piterman, N.; Smolka, S. A. Vol. 7795. Lecture Notes in Computer Science. Springer, 2013, pp. 641–643.

[HR92]　　　　Horwitz, S.; Reps, T. W.: The Use of Program Dependence Graphs in Software Engineering. In: *Proceedings of the 14th International Conference on Software Engineering*. Ed. by Montgomery, T.; Clarke, L. A.; Ghezzi, C. ACM Press, 1992, pp. 392–411.

[LD22]　　　　Leeson, W.; Dwyer, M. B.: Graves-CPA: A Graph-Attention Verifier Selector (Competition Contribution). In: *Tools and Algorithms for the Construction and Analysis of SystemsTACAS, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS, Part II*. Ed. by Fisman, D.; Rosu, G. Vol. 13244. Lecture Notes in Computer Science. Springer, 2022, pp. 440–445.

[MB08]　　　　Moura, L. M. de; Bjørner, N. S.: Z3: An Efficient SMT Solver. In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS Proceedings*. Ed. by Ramakrishnan, C. R.; Rehof, J. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340.

[MH16]　　　　Melnikov, V.; Hüllermeier, E.: Learning to Aggregate Using Uninorms. In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD, Proceedings, Part II*. Ed. by Frasconi, P.; Landwehr, N.; Manco, G.; Vreeken, J. Vol. 9852. Lecture Notes in Computer Science. Springer, 2016, pp. 756–771.

[MH19]　　　　Melnikov, V.; Hüllermeier, E.: Learning to Aggregate: Tackling the Aggregation/Disaggregation Problem for OWA. In: *Proceedings of The 11th Asian Conference on Machine Learning, ACML*. Ed. by Lee, W. S.; Suzuki, T. Vol. 101. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1110–1125.

[MLL04]　　　Moon, S.-I.; Lee, K. H.; Lee, D.: Fuzzy branching temporal logic: Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34 (2004), no. 2, pp. 1045–1055

[MW15]　　　Mohr, F.; Walther, S.: Template-Based Generation of Semantic Services. In: *ICSR*. Ed. by Schaefer, I.; Stamelos, I. Vol. 8919. Lecture Notes in Computer Science. Springer, 2015, pp. 188–203.

[PTF+21]　　　Pellegrini, G.; Tibo, A.; Frasconi, P.; Passerini, A.; Jaeger, M.: Learning Aggregation Functions. In: *Proc. IJCAI, Thirtieth International Joint Conference on Artificial Intelligence*. 2021

[RHJW20]　　Richter, C.; Hüllermeier, E.; Jakobs, M.; Wehrheim, H.: Algorithm selection for software validation based on graph kernels. In: *Autom. Softw. Eng.* 27 (2020), no. 1, pp. 153–186.

[RW19]　　　　Richter, C.; Wehrheim, H.: PeSCo: Predicting Sequential Combinations of Verifiers - (Competition Contribution). In: *Tools and Algorithms for the Construction and Analysis of Systems TACAS: TOOLympics, Part III*. Ed. by Beyer, D.; Huisman, M.; Kordon, F.; Steffen, B. Vol. 11429. Lecture Notes in Computer Science. Springer, 2019, pp. 229–233.

[RW20]   RICHTER, C.; WEHRHEIM, H.: Attend and Represent: A Novel View on Algorithm Selection for Software Verification. In: *35th IEEE/ACM International Conference on Automated Software Engineering, ASE*. IEEE, 2020, pp. 1016–1028.

[SDNW21]   SHARMA, A.; DEMIR, C.; NGOMO, A. N.; WEHRHEIM, H.: MLCHECK- Property-Driven Testing of Machine Learning Classifiers. In: *20th IEEE International Conference on Machine Learning and Applications, ICMLA*. Ed. by WANI, M. A.; SETHI, I. K.; SHI, W.; QU, G.; RAICU, D. S.; JIN, R. IEEE, 2021, pp. 738–745.

[SMHW22]   SHARMA, A.; MELNIKOV, V.; HÜLLERMEIER, E.; WEHRHEIM, H.: Property-Driven Testing of Black-Box Functions. In: *10th IEEE/ACM International Conference on Formal Methods in Software Engineering, FormaliSE@ICSE*. ACM, 2022, pp. 113–123.

[SS02]   SCHÖLKOPF, B.; SMOLA, A. J.: *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002.

[SW20]   SHARMA, A.; WEHRHEIM, H.: Higher income, larger loan? monotonicity testing of machine learning models. In: *ISSTA '20: 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. Ed. by KHURSHID, S.; PASAREANU, C. S. ACM, 2020, pp. 200–210.

[TKK+14]   TULSIAN, V.; KANADE, A.; KUMAR, R.; LAL, A.; NORI, A. V.: MUX: algorithm selection for software model checkers. In: *11th Working Conference on Mining Software Repositories, MSR Proceedings*. Ed. by DEVANBU, P. T.; KIM, S.; PINZGER, M. ACM, 2014, pp. 132–141.

[UAC18]   UDESHI, S.; ARORA, P.; CHATTOPADHYAY, S.: Automated directed fairness testing. In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE*. Ed. by HUCHARD, M.; KÄSTNER, C.; FRASER, G. ACM, 2018, pp. 98–108.

[WW13]   WALTHER, S.; WEHRHEIM, H.: Knowledge-Based Verification of Service Compositions - An SMT Approach. In: *2013 18th International Conference on Engineering of Complex Computer Systems*. IEEE Computer Society, 2013, pp. 24–32.

[WW14]   WALTHER, S.; WEHRHEIM, H.: Verified Service Compositions by Template-Based Construction. In: *FACS*. Ed. by LANESE, I.; MADELAINE, E. Vol. 8997. Lecture Notes in Computer Science. Springer, 2014, pp. 31–48.

[WW16]   WALTHER, S.; WEHRHEIM, H.: On-the-fly construction of provably correct service compositions - templates and proofs. In: *Sci. Comput. Program.* 127 (2016), pp. 2–23.

[ZHML22]   ZHANG, J. M.; HARMAN, M.; MA, L.; LIU, Y.: Machine Learning Testing: Survey, Landscapes and Horizons. In: *IEEE Trans. Software Eng.* 48 (2022), no. 2, pp. 1–36.

[ZWS+20]   ZHANG, P.; WANG, J.; SUN, J.; DONG, G.; WANG, X.; WANG, X.; DONG, J. S.; DAI, T.: White-box fairness testing through adversarial sampling. In: *ICSE '20: 42nd International Conference on Software Engineering*. Ed. by ROTHERMEL, G.; BAE, D. ACM, 2020, pp. 949–960.

Subproject B4:
Verifying Software and Reconfigurable Hardware Services

Eric Bodden[1], Marie-Christine Jakobs[3], Felix Pauck[1], Marco Platzner[1],
Philipp Schubert[1], Heike Wehrheim[2]

1 Department of Computer Science, Paderborn University,
Paderborn, Germany

2 Department of Computer Science, Oldenburg University,
Oldenburg, Germany

3 Department of Computer Science, TU Darmstadt,
Darmstadt, Germany

1 Introduction

Subproject B4 focuses on designing quality assurance measures for single services that are (i) purchased, (ii) composed into service compositions, and (iii) directly employed at runtime in an on-the-fly manner. These measures must allow one to check if an acquired IT service actually fulfills the properties as promised by the service provider. The techniques for ensuring the quality of services must further enable their users to quickly check whether the desired properties hold without forcing them to expensively analyze the service and verify the properties themselves. In this subproject, we consider service providers that assemble compositions and compute centers that execute services as users. The target of the quality assurance measures are individual IT services that are offered in an OTF market. Since services might be implemented in software or synthesized in reconfigurable hardware components, measures to check both must be created.

For example, a service composition used for image recognition may rely on a software or hardware service implementing a filter that is used as an image preprocessor. Hence, it must be ensured that this preprocessing service is safe to use. To this regard, *safety* stands for the property that no error location can be reached.

To reach these goals, Subproject B4 has proposed *proof-carrying services*. Proof-carrying services come with a proof in form of a certificate that allows its users to efficiently check whether the certificate and therefore the properties that the service claims to hold are valid or not, instead of requiring them to extensively analyze and compute the proof for the target service. The idea is to shift the computational expense of verifying the desired properties of an IT service to its respective provider. With respect to software services, the technique implemented for creating and checking certificates is called *proof-carrying code (PCC)* or, in case of reconfigurable hardware components, *proof-carrying hardware*

eric.bodden@uni-paderborn.de (Eric Bodden), jakobs@cs.tu-darmstadt.de (Marie-Christine Jakobs),
felix.pauck@uni-paderborn.de (Felix Pauck), platzner@uni-paderborn.de (Marco Platzner),
philipp.schubert@uni-paderborn.de (Philipp Schubert), heike.wehrheim@uni-oldenburg.de (Heike
Wehrheim)

(PCH). Besides PCC, the *Programs-from-Proofs (PfP)* technique has been proposed, which follows the same goal. However, in contrast to PCC, PfP does not attach certificates or proofs to a service, but instead uses the proof to transform the program (service) into an equivalent program for which the properties of interest can be verified more easily—the service provider thus still verifies the original program whereas the user only has to verify the transformed program. PfP is in depth described as one of the subproject's selected topics in Section 2.1.

In the area of PCH, we have proposed techniques to verify functional and non-functional properties. As an example, in Section 2.2 we elaborate on certifying memory access monitors for reconfigurable hardware systems. In such systems, different modules need to access shared memory, and predefined static or dynamic memory access patterns describe legal access sequences. A memory access monitor is a runtime module that captures these patterns and blocks illegal accesses. Certifying such monitors instead of the complete modules greatly reduces the required computational effort.

Under the term *hardware/software-co-verification (HW/SW-co-verification)* Subproject B4 has developed techniques that pair PCC and PCH. These techniques target services or programs that use so-called *custom instructions* to trigger reconfigurable hardware components. In order to pair PCC with PCH, pre- and postconditions are computed during software verification, such that the hardware verification must assure that these conditions hold. These conditions become part of the certificate and, hence, must only be computed by the service provider, which further unburdens the user. The selected topic presented in Section 2.3 provides more information about HW/SW-co-verification.

In both areas, software (PCC) and hardware (PCH), only safety properties were initially considered. Later on in the project, the focus shifted to the more challenging—with respect to verification/analysis complexity—security properties. This shift required the design of novel techniques as well as the implementation of new frameworks and tools. In Section 2.4, we present the novel PHASAR framework that we developed as part of Subproject B4. PHASAR allows one to statically analyze software written in languages from the C family. We use it to design and prototype new analysis algorithms and strategies to effectively compute safety *and* security properties (and their proofs) for the target services.

Existing mature static analysis tools were also used to create certificates for security properties. These tools usually provide no proof; hence, the quality of the certificate relies on the quality or accuracy of the analysis. Therefore, instruments to determine the accuracy of analyses become indispensable. Consequently, in Section 2.5 we take a closer look at benchmarking software analyses.

2   Selected Research Topics

2.1   Programs-from-Proofs

The goal of Subproject B4 is to provide approaches that let consumers (users) of software or hardware services efficiently and automatically check whether a service ensures the desired properties. One means to achieve this goal is to apply the principle of proof-carrying code (PCC). To achieve efficient checking, PCC relocates the major workload of
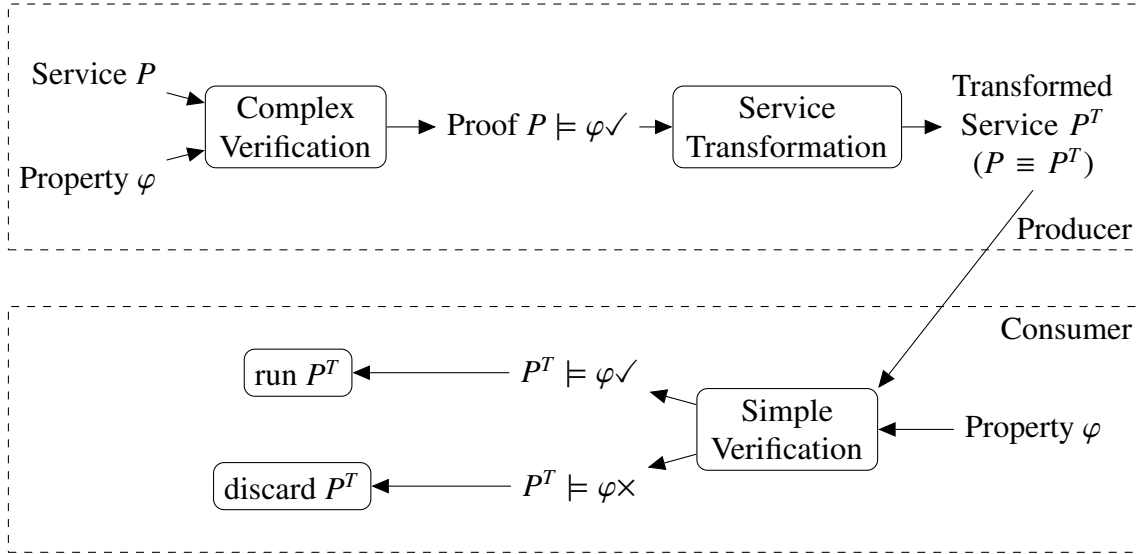
Figure 33: *Generic workflow of the Programs-from-Proofs approach.*

checking: namely, performing the proof that the service ensures the desired properties, to the service producer. The producer then attaches the generated proof in form of a certificate to the service. Hence, the consumer only needs to check whether the certificate attests that the service ensures the desired properties, which is typically assumed to be more efficient than proof generation. Several PCC instances for various properties and analysis techniques have been suggested, some relying on mathematical proofs and others on more general concepts of proofs, such as abstract state spaces. However, often these approaches suffer from large certificates. Furthermore, consumers are bound to specific validation approaches often tailored to the type of certificate and cannot apply existing verification technology. In addition, proof generation is not automatic for all PCC instances.

To overcome these issues, Subproject B4 has proposed an alternative principle, named Programs-from-Proofs (PfP). Like PCC, PfP is a generic principle that still forces the service producer to perform the work-intensive part of proof generation. However, it goes without certificates and lets the consumer employ existing, but relatively efficient verification techniques, such as dataflow analyses instead of specific validation techniques, for example. To achieve this, PfP uses the insights that the structure of a software service (i.e., program) can heavily influence the complexity of verification but that many proofs, in particular proofs that model the (abstract) state space of a (software) service, restructure the service such that its verification becomes simpler. More concretely, PfP employs the proof to transform the (software) service into a different, but behaviorally equivalent and property preserving (software) service that is easier to verify.

Figure 33 shows the generic workflow of the PfP approach, which we explain in more detail in the following.

1. Initially, the producer verifies (software) service P with respect to property φ, applying a potentially complex and costly verification. During the complex verification, many PfP instances use a combination of a computational expensive, incremental analysis and a cheap analysis. The cheap analysis is responsible for checking the property while the main purpose of the expensive analysis is to restructure the (abstract) state space, i.e., to restructure the paths of the analyzed (software) services by

unfolding loops, avoiding reintegration of branches, excluding infeasible execution paths, etc., such that the cheap analysis succeeds in property checking. To achieve the necessary restructuring, the expensive analysis often performs the restructuring incrementally based on the failed proof attempts of the cheap analysis.

2. After the producer's verification attempt succeedes, the producer uses the proof to automatically transfer the restructuring that was done for proving the property to the service. Thereby, it is important that the transformation (1) does not change the service's (functional) behavior, (2) keeps the validity of the property, and (3) ensures simple verification of the property on the transformed (software) service P^T. All PfP instances Subproject B4 has developed focus on proofs in form of abstract reachability graphs (ARGs). An ARG is a representation of the abstract state space of a (software) service. Important for the PfP instances is that all ARG paths correspond to syntactic paths in the analyzed (software) service and that all syntactic paths that are also semantically feasible (i.e., the executable paths) are represented in the ARG. However, an ARG and the (software) service likely structure paths differently and the ARG may contain less infeasible syntactic paths. All those differences allowed the cheap analysis component to prove the validity of the property. The ARG characteristics mentioned above are the reasons why the PfP instances, which Subproject B4 has developed, all translate the ARG, in particular its paths with their structure, into a (software) service, which becomes the transformed service P^T delivered to the consumer. Furthermore, these characteristics allow one to verify the desired behavioral equivalence of the (software) service before and after transformation.

3. Once the consumer has received the transformed service P^T, he or she performs a simple verification of the transformed service P^T to efficiently and automatically check whether a service ensures the desired property φ. If the complex verification consisted of a combination of cheap and expensive analysis as described above, the simple verification typically applies (a variant of) the cheap analysis technique, although the consumer might use a different implementation of the cheap analysis. Our PfP instances even allow the cheap analysis to become path-insensitive. Typically, our instances each use a variant of the respective cheap analysis that performs an efficient, flow-sensitive dataflow analysis. The reason is that any path sensitivity that the cheap analysis contributed during complex verification is also incorporated in the ARG structure and, thus, in the transformed (software) service. To prevent the consumer from harm, the simple verification must be tamper-proof, i.e., it must detect any tampering of the process that invalidates the desired property on the received service, e.g., deviations in properties, invalid producer proofs, incorrect transformation, or changes to the transformed service during delivery. Hence, the simple verification must be sound, i.e., it must ensure that only services that fulfill the desired property are verified successfully. Since soundness is typically guaranteed by the simple verification technique itself, we focused on showing successful consumer verification in a tamper-free PfP workflow. More concretely, for the PfP instances Subproject B4 has developed, we have proven that the simple verification will succeed if the complex and simple verification consider the same property, the complex verification has succeeded, and the simple verification verifies the services computed by the transformation based on the proof generated by the complex verification.

4. Depending on the outcome of the simple verification, the consumer lastly either runs the transformed service in case of a successful verification or otherwise discards the service.

Our proof-of-concept instance for PfP [WSW13] has addressed typestate properties, protocol-like properties enhancing types with information about their state, and has introduced the idea to transform ARGs into services (programs). Its complex verification combines predicate model checking and a typestate analysis, while the simple verification performs a pure typestate dataflow analysis. A typestate analysis allows to decide whether certain operations are possible with respect to the typestate of a variable. For example, an integer variable may be in the typestate uninitialized, demanding that it is initialized before it is used. Subsequent PfP instances [JW15; JW17] have extended the supported types of analyses and properties, but reuse the idea of ARG to service (program) transformation. Furthermore, we have used the software analysis framework CPAchecker [BK11], a tool that supports configurable program analysis, to implement our PfP instances. While we have reused CPAchecker's existing analyses and its possibility to combine analyses to realize the complex and simple verification, we have integrated the ARG to service (program) transformation into CPAchecker. Practical evaluations of our PfP instances with CPAchecker have shown that the consumer's simple verification is indeed significantly more efficient in terms of runtime and memory usage than the producer's complex verification. Also, PfP is often more efficient than existing PCC approaches applicable to configurable program analyses.

The PfP approach here makes a first essential contribution in the range of the proof procedures. As described before, PfP addresses the problem that proofs stored in the proof-carrying code method are usually very large and therefore inefficient to handle. It could be shown that this can succeed by means of PfP to embed the proof quasi partially directly into the structure of the program which can be analyzed. Thereby, the size of the proof is reduced and nevertheless the possibility of the efficient proof examination by the user remains. As a result, PfP thus allows for an often more efficient examination of the necessary evidence and a more efficient transfer of this evidence to the user. However, another advantage of PfP over PCC is also the reduction in *trusted base*: In PCC, the user must trust the verification procedure, which itself is often relatively complex (albeit runtime efficient). In PfP, however, this checking procedure corresponds to a relatively simple data flow analysis, which should increase confidence that this procedure is error-free. PfP thus increases confidence in the overall security of the corresponding services.

2.2 Proof-Carrying Hardware

Proof-carrying hardware (PCH) was first proposed by Drzevitzky et al. [DKP09; DKP10] as the reconfigurable hardware equivalent of PCC. The PCH concept distinguishes a circuit producer (e.g., a design center) and a consumer, e.g., a data center operating a reconfigurable computer or an embedded system based on a reconfigurable system-on-chip. The consumer loads and executes reconfigurable hardware modules that were created by the producer. Additionally, the consumer specifies a security property that the modules need to fulfill and, before loading, requires formal proofs of the properties. It is the task of the producer to generate not only the modules but also the proofs and transmit both to

the consumer. The consumer will verify that the proofs are correct and actually belong to the modules. In PCH, the compound of module implementation and the proof have been denoted as a *proof-carrying bitstream*.

An important security property for reconfigurable hardware systems pertains to memory access policies. The density of today's reconfigurable hardware devices allows for implementing reconfigurable systems with a large number of modules or cores, respectively. Through dynamic or even partial reconfiguration, modules can be loaded on demand, increasing flexibility. Several modules that access the same physical memory need to adhere to a specified policy governing their access patterns. A simple static policy, for example, is to enforce that each core can only access its own segment of the memory. There are, however, more involved policies in use when it comes to intended sharing of data between cores, the handling of conflict-of-interest classes, or different security levels. Huffmire et al. [HSKL08] introduced a *monitoring-based approach* to ensure memory access security. They presented a formal language and a compilation tool flow that allows a designer to specify a memory access policy and generate a circuit for a so-called memory access monitor. All modules' memory accesses have to be routed through the monitor, enabling the monitor to block any memory access that violates the policy at runtime.

We guarantee memory access security in the strength of formal verification by bringing together the monitoring approach of Huffmire et al. with the proof-carrying hardware concept. The consumer operates a reconfigurable resource where several cores access shared memory and memory accesses are routed through a memory access monitor that implements a predefined memory access policy. The policy can change during runtime to reflect different applications and security requirements. The consumer receives a new monitor together with a proof of its functional correctness, verifies the proof and, in case of success, partially reconfigures the monitor.

Our tool flow starts with the consumer that uses behavioral Verilog to specify the memory access policy. The producer receives the design specification and synthesizes it into an FPGA bitstream, using the tools of Huffmire et al. and, subsequently, VTR for Verilog synthesis and place & route. After that, the producer re-extracts the logic function from the bitstream and, together with the original design specification, computes the miter function. The miter function is shown in Figure 34 and is constructed such that the output of the miter, i.e., the error flag, can only be 1 if the specification and implementation differ for at least one input vector. For proving functional equivalence for combinational circuits, it is thus sufficient to prove unsatisfiability of the miter. We use ABC to construct a miter in conjunctive normal form and the SAT solver PicoSAT to prove unsatisfiability. PicoSAT also generates a proof trace that, together with the bitstream, forms the proof-carrying bitstream.

Dynamic memory access policies lead to sequential monitor circuits. Thus, we extended the concept and tool flow to also work with sequential miters using bounded sequential equivalence checking [WDP14]. A sequential miter circuit is unrolled for a specified number n of time frames, resulting in n copies of the circuit that are connected at their flip flops. Every time frame represents one clock cycle, and we can change the primary inputs and observe the primary outputs at every individual cycle. The miter construction then compares all outputs in each time frame and the flip flop signals of the last frame, and raises the error flag if there is a deviation somewhere. As we have to choose a specific amount of unrolling time frames, we observe that the compiled monitors are essentially
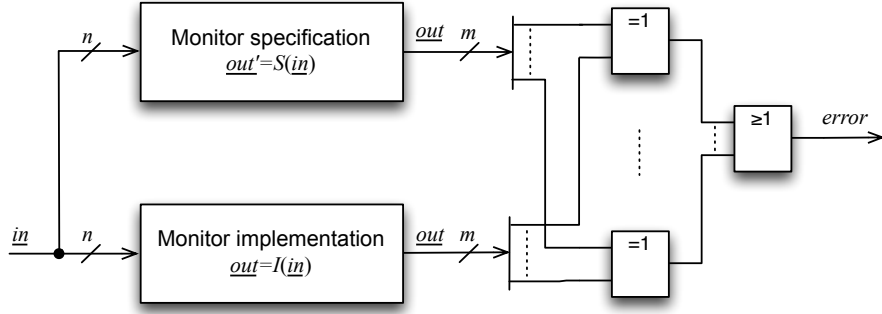
Figure 34: *Miter $M(S(x), I(x))$ for proving the functional equivalence of specification S and implementation I, (taken from [WDP14]).*

state machines, and their internal transitions only depend on their current state and the new input. Suppose there is an input sequence i that satisfies the miter function, i.e., it leads to different outputs for the implemented circuit and the specification, and the corresponding state transition path of the state machine contains cycles. Then the input sequence i', which leaves out all state cycles of i, is also a valid input sequence and it also satisfies the miter. If the miter is thus provably unsatisfiable for all maximum length-cycle free state transition paths, it is unsatisfiable for all input sequences of all lengths. Hence, we can simply use a number n of unrolling frames larger than the number s of automaton states to ensure that every cycle-free sequence has been considered.

The consumer receives the bitstream for the monitor circuit together with the proof trace for unsatisfiability. In a first step, the consumer also extracts the monitor's logic function from the bitstream and forms a miter in conjunctive normal form in the same way as the producer, but with the original specification. The so-created miter is compared to the miter sent by the producer, which is part of the proof trace. If the miters do not match, then the proof is not based on the desired functionality and the monitor is refused. If the miters match, the consumer verifies the proof by checking each reduction step in the proof trace until an empty clause results. Only then, is the implementation shown to adhere to the security property and the monitor accepted.

To demonstrate the capability of our proposed approach for ensuring memory security, we built a prototypical system. As platform we chose a ZedBoard containing a Xilinx Zynq-7000 system-on-a-chip with a dual ARM Cortex-A9, and 512 MB RAM. Our prototype architecture embeds a *virtual FPGA overlay* into a reconfigurable system as shown in Figure 35. We use a virtual FPGA since we need to be able to interpret the transmitted configuration bitstream for the memory access monitor. FPGA vendors typically do not share the necessary information, and reverse engineering the bitstream or additionally transmitting and interpreting low-level circuit descriptions such as Xilinx XDL are extremely tedious processes. Virtual FPGAs or FPGA overlay architectures have become increasingly popular in the last years for a number of reasons. They provide a means to implement portable circuits, bring partial reconfiguration capabilities to FPGAs that have no native support of this feature, achieve fast configuration rates, prototype coarse grained arrays, or be able to implement circuits created with open source tool flows such as VTR on real FPGAs.

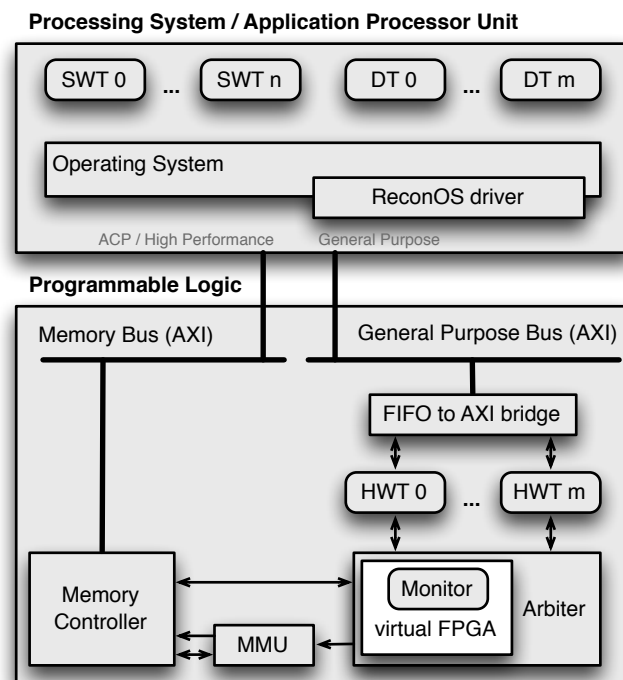We leverage the virtual FPGA overlay ZUMA and embed it into ReconOS [LP09]. Re-

Processing System / Application Processor Unit

| SWT 0 | ... | SWT n | | DT 0 | ... | DT m |

Operating System

ReconOS driver

ACP / High Performance   General Purpose

Programmable Logic

Memory Bus (AXI)   General Purpose Bus (AXI)

FIFO to AXI bridge

HWT 0   ...   HWT m

Memory Controller

MMU

Monitor
virtual FPGA   Arbiter

Figure 35: *Xilinx Zynq version of ReconOS, with n + 1 software threads (SWT), m + 1 hardware threads (HWT), their m + 1 delegate threads (DT), and an arbiter including a memory monitor in the memory access path of the HWTs (taken from [WDP14]).*
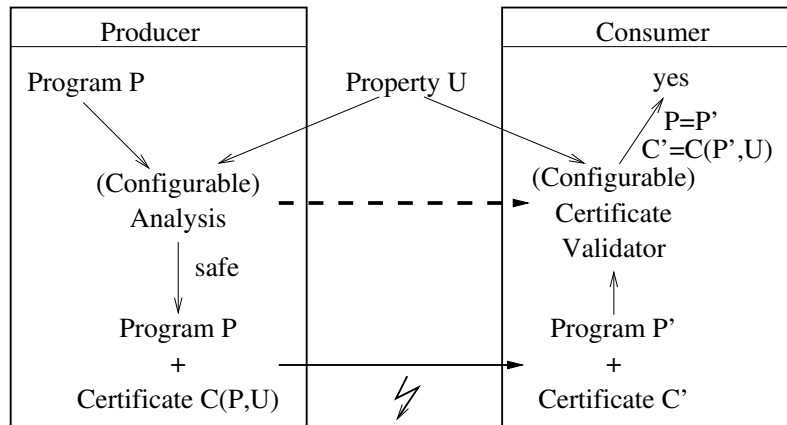
Figure 36: *Certification generation and validation.*

conOS is an execution environment for hybrid hardware/software systems featuring a multithreaded programming model that allows for regular software threads as well as hardware threads. The use of ReconOS enables us to use a mature, Linux-based infrastructure for implementing hardware/software systems, including a CPU core, memory controller, peripherals and a standard software operating system. As shown in Figure 35, we have modified the ReconOS arbiter in the memory access path of the hardware threads to include the memory access monitor. The access monitor itself is implemented in our ZUMA virtual FPGA overlay. For the inputs, the arbiter provides the monitor the virtual memory address, the type of the request (read or write) and its source, the hardware thread identifier.

We further presented a series of experiments to investigate different aspects of our approach and prototype. The experiments showed that the approach is feasible and can secure static and dynamic memory access policies of different complexities. With 61.84% to 90.53% of the overall workload, depending on the memory access policy, the producer clearly bears the computational burden of establishing the consumer's trust in the module. As expected, the overlay comes with rather high area and delay overheads. The reduction of these overheads was also addressed.

2.3   Proof-Carrying Code and Its Relation to Proof-Carrying Hardware

The core principle underlying the work of Subproject B4 was to enable on-the-fly checking of service correctness by attaching proofs as witnesses to the correctness of both software and hardware. Here, we briefly explain our technique of proof-carrying code (software verification) and its integration with hardware verification.

For proof-carrying code we employ analysis and verification techniques that can formally prove the validity of properties in software programs. Hence, we can employ the proofs as a form of *certificate* to a service's correctness. The basic principle of proof-carrying code is the idea that the generation of certificates (on the side of the service producer) can be time-consuming while its validation (on the side of the consumer) should be easy. Figure 36 depicts this basic scheme. The producer develops a program (service) P, which

should adhere to property (requirement) U. The producer is supposed to carry out the costly analysis (proving the holding of property U on P). The outcome of the analysis, more specifically the correctness proof, is then attached to the program in the form of a certificate. When a consumer wants to use this service, it retrieves program and certificate from some repository. Our assumption here is, however, that neither producer nor storage in repositories can be fully trusted. Thus, the consumer might actually receive a slightly different program P' or a slightly modified certificate C'. Our technique enables the consumer to quickly validate whether the certificate still fits to the program and thereby whether the received program P' meets the intended requirement U.

Instead of developing a certification technique per property or per class of properties, we have investigated the generation of certificates for arbitrary properties [JW14] via a *configurable certification process*. Our generic approach builds on an existing framework for configurable program analysis with tool support in the form of CPAchecker [BK11]. CPAchecker executes an analysis *meta algorithm* generating a (structured) abstract reachability set of a given program. The meta algorithm can be steered by a number of user-supplied inputs (e.g., telling CPAchecker when to stop the analysis and when to merge states). This presents a way of uniting different program analysis techniques, ranging from data-flow analyses, to computing abstract information for control flow graphs, to model checking, computing a tree-like abstract structure. The generated reach set is then subject to property checking.

For the certification process, we use the—anyway generated—reach set as certificate. Similar to the analysis, we develop a generic configurable certificate validation framework with a corresponding meta algorithm for certificate checking. In addition, we provide a way of (in a large number of cases automatically) generating the configuration of the certificate validation from a given configuration of the analysis. Our approach is tamper-proof in that the certificate validator only outputs "yes" if the program P remains unchanged ($P = P'$) and the obtained (and possibly corrupted) certificate C' is a valid certificate for the program P with respect to a desired property U. We have implemented our technique within the CPAchecker framework, and evaluated it on a number of different analysis techniques. For all of these, certificate validation is faster than analysis. We proved soundness of all of our techniques, i.e., we have shown them to be tamper-free.

To connect to the certification on the hardware level, we have studied how software certificates relate to the underlying hardware used for execution. Software analyses typically rely on the correctness of the processor hardware executing the program. More specifically, the strongest postcondition computation used to determine the successor state of a given state for a program statement assumes that the processor correctly implements the statement's semantics. Certificate validation heavily employs the strongest postcondition computations. This assumption of correct hardware is certainly valid for standard processors, since they undergo extensive simulation, testing, and partly also formal verification processes. However, during the last years processors with so-called custom instruction (CI) set extensions became popular, which challenge this correctness assumption. Customized instructions map a part of an application's data flow graph to specialized functional units in the processor pipeline in order to improve performance and/or energy efficiency.

In [JPWW14], we have presented a novel formal approach for software/hardware co-verification, in particular for processors with custom instruction set extensions. It (partially) employs the certificate computed by the software analysis to derive requirements on the
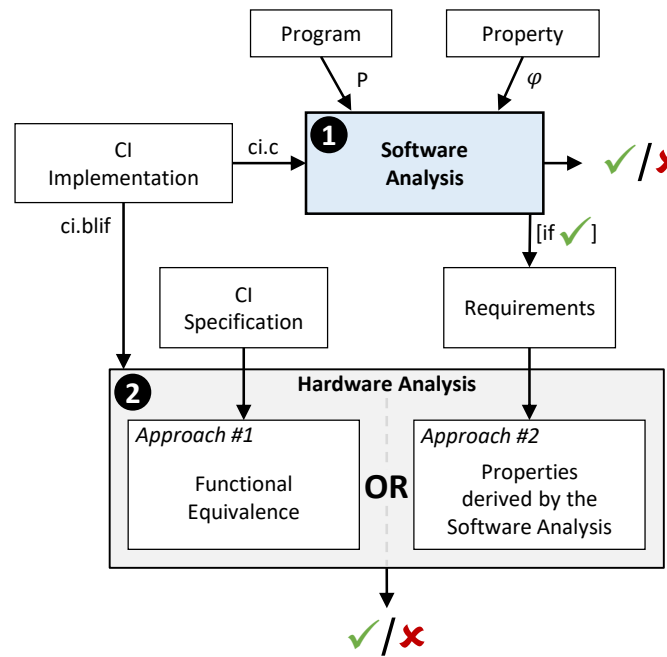
Figure 37: *Overview of hardware-software co-certification.*

hardware. These requirements then need to be validated in order for the software analysis to produce trustworthy results. Figure 37 gives an overview of our approach.

We have studied two different approaches for integrating software and hardware analyses that differ in what needs to be verified on the hardware side. Our first approach proves functional equivalence between the specification and the implementation of a custom instruction, e.g., that an integer adder is actually adding integer values. While proving equivalence is potentially the most runtime-consuming approach, it is also the most powerful, as it inherently covers all behavioral properties of the custom instruction on which software analyses could rely. Our second approach ties together software and hardware analyses more closely by exploiting the abstract state space of the program generated during verification to identify the specific properties of the individual program statements the software analysis has actually used during verification. These properties become *requirements* on the hardware. We thereby tailor the hardware verification exactly to the needs of the software analysis, hoping to avoid unnecessarily complex and runtime-consuming hardware verification.

We have built a toolchain automating all the steps of our approach, which are (1) the software analysis computing requirements on the hardware via the use of a verification tool plus information about the custom instructions, (2) the hardware analysis synthesizing property checkers from requirement and custom instruction specification, and (3) a SAT solver for checking satisfiability of the custom instruction implementation together with the property checker. We have evaluated our technique on different custom instructions occurring in programs using several software analyses for requirements extraction. As a main result from our experimentation, we can conclude that while tailoring the hardware verification more to the concrete needs of the software analysis indeed generally results in lower computational effort, neither approach is superior for all cases.

2.4 Static Analysis with PHASAR

Another selected topic of Subproject B4 is embodied in the genesis and development of the meanwhile well-known PHASAR [SHB19] project for static program analysis. PHASAR is a modular static analysis framework targeting the C and C++ programming languages and has been built on top of LLVMto account for the lack of general infrastructure for the analysis of such programs. PHASAR's infrastructure allows one to quickly draft prototypes for new program analyzers, novel algorithms and analysis strategies, and also allows for their evaluation.

We have built several novel analysis approaches on top of PHASAR, which we present in the following paragraphs.

C/C++ languages are often used for projects that require a direct interface with operating systems or hardware components. They offer control to programmers for creating efficient programs, but also require correct usage to avoid bugs or security issues. Compilers such as GCC and Clang and additional tools, such as Cppcheck and Clang Static Analyzer, aid in creating secure software. However, they often provide only simple checks or have a large number of false or missed warnings due to imprecise analysis. For Java programs, program-analysis frameworks such as Soot, WALA, and Doop provide more precise dataflow analysis. This type of implementation was not available for C/C++. This is where PHASAR came in, a novel program-analysis framework designed for LLVM infrastructure. It can be used for dataflow problems, call-graph construction, and points-to information. PHASAR is intended for static analysis and complements LLVM toolchain features. Some parts may be used as a compiler pass.

C/C++ programs can represent an entire software product line using static conditionals called features. Traditional static analysis techniques cannot be applied to software product lines directly, because the process of generating and analyzing all software products becomes prohibitively expensive due to the possibly exponential number of software products. To solve this problem, VARALYZER, a family-based approach was developed, which analyzes a software product line as a whole. VARALYZER transforms preprocessor directives into ordinary C code using a configuration-aware type checker. It supports not just analyses encoded in IFDS but also those encoded in interprocedural distributive environments (IDE). VARALYZER outputs the fully context- and flow-sensitive dataflow facts along with a feature constraint describing the product configurations for which they hold. This allows developers to find bugs and vulnerabilities much earlier in the development process, when a preprocessor has not yet even been applied, for instance, in a version-control system. The effectiveness of VARALYZER has been evaluated using a typestate analysis that checks for the correct usages of OpenSSL's Envelope (EVP) APIs on 95 compilation units. Challenges related to evaluating VARALYZER on full SPLs are detailed in [SGP+22].

MODALYZER [SHB21] is a novel approach that enables the scaling of static analyses on large software projects. The approach involves the pre-computation of summaries for parts of code that do not frequently change, which can be integrated into larger analysis scopes. The summaries can be seen as proofs of the property the client analysis attempts to demonstrate. Whole-program analysis (WPA), which can be memory-intensive and cause runtime problems, can be substituted with intra-procedural analyses that are simple enough to scale, as demonstrated by tools such as Clang-tidy and Cppcheck. However, semantic

program analyses such as shape, typestate, and dataflow analyses require detailed program representations that include the effects of procedure calls, which are impossible to scale if calculated for the entire program.

MODALYZER provides a compositional approach to program analysis that is capable of scaling static context-sensitive, field-sensitive, and flow-sensitive inter-procedural program analysis. This is achieved through the compositional computation of analysis information. The success of the compositional analysis depends on the number of reusable parts of the application, for example, libraries, or parts that do not change from one analysis run to the next. Black Duck's recent study shows that 96% of the applications they scan contain open-source components, and those components now account for, on average, 57% of the code. The application of compositional analysis can accelerate the analysis of applications by reusing analysis results from previous runs, especially as open-source dependencies are updated much less frequently than application code.

Although previous work on compositional program analysis has been limited to certain types of dataflow analysis, MODALYZER provides a mechanism for analysis dependency management for a fully compositional analysis that automates updates whenever new information becomes available that affects existing information. The approach also involves an efficient summary format that is able to persist general data. MODALYZER can potentially scale the analysis of applications by reusing analysis results from previous runs.

Last but not least, INCALYZER was developed to support summarization and reuse of static analysis information for *frequently* changing parts of a program. It assumes that the target project is developed using a version control system and aims at maximizing the reuse of static analysis information computed on a previous revision of the target project that is still valid. Summarization techniques can be used to pre-compute summaries that can be reused while analyzing the actual application code and may decrease the analysis time by a large factor. Tree-adjoining languages and Dyck context-free language reachability can help to increase the number of useful summaries. Incremental analysis can improve scalability for frequently changing code, as changes made to a program are usually small and thus should only cause invalidation of a small amount of the analysis results. Existing incremental static analysis techniques ignore the information provided by version control systems (VCS) and are only concerned with dataflow information.

Contrary to the REVISER approach, which only considers the dataflow parts of a client analysis for its incremental analysis and computes the code delta based on the inter-procedural control-flow graphs, INCALYZER makes the complete client analysis stack (control-flow, callgraph, points-to, type-hierarchy and dataflow information) incremental and uses VCS information to obtain the code delta directly. If INCALYZER recognizes that a code change has no impact on the semantics of the program while producing commit-annotated IR, no reanalysis is performed on the IR. INCALYZER has great potential to allow developers to check-in persisted static analysis results directly to the VCS managed code repository for each commit of a project which are then both kept in sync throughout the continuous integration development of the project. This has the advantage that each revision only needs to be analyzed once. Any developer can check out a code revision accompanied by its respective up-to-date analysis results, allowing them to check and reuse them for incremental analysis locally. This allows static analysis information for each commit to be viewed as "certificate" which can be checked instantaneously for each given commit, according to the precision and capabilities of the underlying client analysis, of course. One

may even bind those "certificates" to the code, e.g., using cryptographic hashing, to avoid accidental or intentional manipulation.

2.5　Benchmarking with REPRODROID

The number of research communities fostering open science is steadily increasing. For instance, the software engineering community has turned artifact evaluations from a rarity into a standard. Funding agencies nowadays join this effort by rewarding the availability of open science artifacts. For these reasons, instruments to drive reproducible evaluations have become more important and needed than ever before. Building such instruments, in particular in the context of on-the-fly computing, proves to be challenging, since the market and its ecosystem must be available and accessible. With REPRODROID [PBW18], a framework that allows to create or adapt benchmarks so that these can be executed and evaluated automatically, we have proposed such an evaluation instrument for Android taint analysis. We have used REPRODROID to evaluate whether six "Android taint analysis tools keep their promises" [PBW18], to create, execute and evaluate a real-world benchmark [LPP+22] and to evaluate cooperative analyses [PW19].

Android *taint analyses* track the flow of sensitive data throughout one or multiple apps. Whenever sensitive information is accessed via a private *source*, it is marked as tainted and tracked through the app's data (and control) flow. If tainted data reaches a public *sink*, a data leak is reported in form of a *taint flow* that stretches from source to sink. We differentiate *intra-app* taint flows inside a single app from *inter-app* taint flows between apps.

To evaluate taint analysis tools, benchmarks are usually employed. A *benchmark*, in this context, consists of two parts: a set of apps and its *ground truth*, which is a complete list of all taint flows occurring in these apps. Since it is often difficult to determine whether a ground truth is correct or complete, micro benchmarks are often used. *Micro benchmarks* consist of tiny apps that were only implemented for benchmarking purposes. Each micro benchmark app usually implements only a single taint flow that uses or exploits a specific Android or programming language feature. Hence, the ground truth can be defined by documenting this specific taint flow only.

In the past, a benchmark's ground truth was often described in natural language, which allowed different interpretations and ultimately led to irreproducible results. REPRODROID uses the Android app analysis query language (AQL) [PBW18; PW19] to precisely specify a benchmark's ground truth and to interact with arbitrary Android taint analysis tools. Figure 38 provides an overview of REPRODROID's toolchain. First, the benchmark refinement and execution wizard (BREW) takes a set of apps as input. During Step 1, the sources and sinks that occur in these apps are identified. BREW allows to automate this process by automatically selecting sources and sinks which are specified in a configurable list. Such lists are typically used by taint analysis tools to identify the respective statements. Furthermore, for each pair of source and sink that belongs to the same benchmark case, it is specified whether it describes an expected or a not-expected taint flow. While an *expected* taint flow should be found by an analysis, a *not-expected* taint flow should explicitly not be found—finding it is considered to be a false positive result. Once the ground truth is fully described in BREW, the benchmark is ready to be executed. To do so, BREW
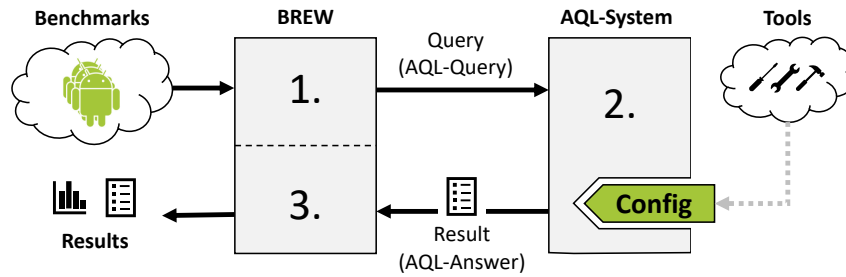
Figure 38: *Sketch of the* REPRODROID *toolchain.*

forwards one AQL query per benchmark case to the next component: namely, the AQL SYSTEM, which performs Step 2 (see Figure 38). As the name suggests, the AQL SYSTEM is the default system for using the AQL. In case of a query asking for taint flows, the AQL SYSTEM looks up a taint analysis tool in its configuration and runs it in order to answer the query. If required, the taint analysis tool's output is converted into the AQL answer format. This answer is replied to BREW, which then compares the answer against the ground truth to finally compute the accuracy metrics precision, recall and F-measure (Step 3). These metrics summarize the benchmark's outcome and allow us to compare the performance of different tools.

In a first study [PBW18], we have used REPRODROID to check the feature and accuracy promises given for six taint analysis tools. For example, it is claimed that FLOWDROID, the most-cited tool, is context-, flow-, field-, object-sensitive and lifecycle-aware and that it achieves certain precision, recall and F-measure scores for the DROIDBENCH benchmark. Additionally, it is claimed that these tools are able to analyze real-world apps—a promise that we have also attempted to validate. In conclusion, we have found that most promises were kept by most tools. However, all of them seemed to struggle in case of real-world scenarios.

Initially, we have used REPRODROID to adapt the most-used (with respect to citations) micro benchmarks for Android taint analyses (DROIDBENCH and ICC-BENCH) such that they can automatically be executed and evaluated to guarantee reproducibility and comparability. Later, the real-world benchmark TAINTBENCH [LPP+22] was created with and for REPRO-DROID. TAINTBENCH comprises 39 malware apps that have been shipped via various app markets. For these 39 apps, 203 expected and 46 not-expected taint flows have been determined manually and specified in REPRODROID. Even though this ground truth is most likely incomplete, through the definition of expected and not-expected taint flows we are still able to evaluate taint analysis tools on this baseline. In the end, TAINTBENCH has allowed us to gain novel and measurable insights that reveal capabilities and inabilities of analyses especially while handling real-world scenarios. Most surprisingly, it has also allowed us to detect regressions between two versions of two state-of-the-art analysis tools (AMANDROID and FLOWDROID) that were not visible using micro benchmarks only.

Combinations of analyses (*cooperative analyses*) can also be evaluated by means of REPRODROID [PW19]. In this case, the AQL is not only used to interact with arbitrary analysis tools but also to steer the cooperation between analysis tools, e.g., how to combine their results. To efficiently execute cooperative analyses, the AQL SYSTEM allows to distribute the execution of different tools onto distinct and distributed AQL SYSTEMS. We have composed four cooperative strategies that overall employed 12 analysis tools in order

to deal with four analysis challenges. One of these strategies, for instance, deals with inter-app communication. This strategy allows to detect taint flows that start in one app and end in another. To do so, a taint analysis tool is queried to find intra-app taint flows and a combination of two additional tools to find inter-app flows. By means of the AQL, these intra- and inter-app flows are stitched together which has ultimately allowed us to detect taint flows across app boundaries. In case of all four challenges (reflection, native code, inter-component, and inter-app communication) significant improvements were able to be achieved through cooperation and measured via ReproDroid.

In the context of on-the-fly computing, cooperative analyses can be interpreted as service compositions themselves, i.e., each analysis tool represents a service, an AQL query describes the service composition, and the AQL System stands for a service provider, whereas another AQL System may take the role of a compute center. In this scope, ReproDroid can be used to determine the quality of services and service compositions. Due to the reproducible nature of benchmarks executed via ReproDroid anyone (consumer or producer) is able to check whether certain properties (e.g., accuracy metrics) are accomplished by a service (composition). Trustworthy and demonstrably accurate (cooperative) analyses can then be used for the "certification" of other services or service compositions.

3   Impact and Outlook

Subproject B4 has worked on various proof-carrying service techniques throughout all three periods of the CRC 901. In the beginning, the fundamentals of proof-carrying code (PCC) and proof-carrying hardware (PCH) have been examined closely, extended, and implemented in first prototypes. The evaluations conducted along the way have already proven the potential of these techniques in the context of on-the-fly computing, i.e., safety properties of services, to be used in service compositions, they could be certified by service providers (producers), and they were less expensively checked by their users (consumers, e.g., compute centers). Next, mainly during the second period, (1) PCC and PCH have been brought together such that software and hardware services interacting with each other can be certified collectively, (2) mature implementations have been developed to extend the field of application, such that more versatile (software and hardware) services and properties, in particular security-related properties, can be analyzed and certified, and (3) techniques to assess the quality of analyses have been researched and implemented. For example, the Phasar analysis and the ReproDroid benchmarking framework were constructed. While approaching the end of the CRC, the benefits of the effort spent so far became measurable not only in terms of more than 100 publications contributed by Subproject B4 but also in terms of available and usable artifacts, which have and will cause impact beyond research.[13] In the following, we present and discuss these benefits in the context of the five selected topics detailed above.

Based on the core scheme of proof-carrying code, we have investigated a number of optimizations and extensions (e.g., Programs-from-Proofs). The goal, for instance, was to provide more compact certificates. We have furthermore studied certification techniques for *hyperproperties*, more specifically for information flow properties [TW18]. Information flow analysis investigates the flow of data in applications, checking in particular for flows

---

[13]`https://ris.uni-paderborn.de/project/12` (19.04.2023)

from private sources to public sinks. Flow- and path-sensitive analyses are, however, often too costly to be performed every time a security-critical application is run. We have proposed a variant of proof-carrying code for information flow security. To this end, we have developed information flow certificates that get attached to programs as well as a method for information flow certificate validation. The technique has also been implemented within the program analysis tool CPAᴄʜᴇᴄᴋᴇʀ [BK11]. Furthermore, we have studied different security policies for information flow and their integration in a certification context [TW18].

Programs-from-Proofs (PfP) represents one of these proof-carrying code (PCC) optimizations for which we have in turn proposed several extensions. The first PfP extension supports reachability properties and any kind of dataflow analysis as cheap analysis. Hence, complex verification becomes a combination of predicate model checking and an arbitrary dataflow analysis, named predicated dataflow analysis, while the simple analysis uses the dataflow analysis alone. Later, a generic PfP framework [JW17] has added support for arbitrary properties expressible as property automaton (including typestate and reachability properties). In addition, the framework allows to combine arbitrary expensive and cheap analyses in the complex verification as long as the cheap analysis solely checks the property, it is at least flow-sensitive, and both analyses are expressible in the framework of configurable program analysis [BK11], which allows to describe arbitrary abstract-interpretation based analyses. The simple analysis then uses the cheap analysis reconfigured as a dataflow analysis. Not only the generic PfP framework but all our PfP instances rely on the existing concept of configurable program analysis to describe the analyses: in particular, complex and simple verification as well as the combination of expensive and cheap analyses. As a last extension we have also adopted the idea of PfP to perform runtime verification with no overhead. These extensions and in particular the generic framework, show that the Programs-from-Proofs technique is highly applicable with respect to various properties and services. In conclusion, due to our research and implementations, the PfP approach has become a usable approach instead of a mostly theoretical concept.

Besides developing proof-carrying hardware (PCH) frameworks for certifying functional equivalence for combinational and sequential circuits, we presented PCH approaches for certifying non-functional security properties such as the worst-case execution time of hardware modules and keeping predefined error bounds for approximated circuits. For the demonstration of the PCH concept, we relied first on abstract FPGAs that could only be simulated, and later on virtual FPGA overlays that allowed us to show the feasibility of PCH on real FPGA hardware. In more recent work, we studied PCH as a tool for detecting hardware trojans in reconfigurable modules and showcased these methods on Lattice FPGAs with their known bitstream formats. Lastly, we want to mention that the proof-carrying hardware term that was introduced in the context of this subproject has been taken up by others [LJM12]. This silently demonstrates the impact of our research conducted in this area.

The concepts of PCC (software) and PCH (hardware) are built on the notion of a *certificate* certifying the correctness of software or hardware with respect to specified requirements. For the software, this is (in our project) a compact version of the abstract reachability graph (ARG) constructed during software verification. On the consumer side, the ARG is checked for two properties: (1) its fit to the program, i.e., whether it is an abstract reachability graph for the program, and (2) its consistency with the requirement, i.e., whether it actually proves

program correctness. While we used these certificates to realize the collaborative analysis of software and hardware services, such certificates have also recently been employed in software verification competitions such as SV-COMP [Bey22]. SV-COMP is an annual competition for software verification mainly targeting C programs. The tools participating in the competition have to determine whether a specified requirement is met or not. In the first case, tools are required to provide *correctness witnesses*, in the latter, *violation witnesses*. The correctness witnesses serve the same purpose as our certificates (and almost take the same form). Witnesses are then also checked for their soundness using so-called witness validators. This usage of certificates in competitions indicates and exemplifies that the concepts also proposed by Subproject B4 are adopted and used by others.

We started the Phasar project in 2016 and made the first version publicly available in 2018 in a full-day workshop at the *39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (PLDI) conference. As of today, the Phasar github repository has achieved 773 stars, was forked 123 times, and has grown far beyond the scope of Subproject B4 as 41 developers from around the globe contributed to the framework.[14] Moreover, each of the mentioned extension of Phasar (ModAlyzer, IncAlyzer, VarAlyzer—see Section 2.4) is accompanied by a research paper that includes extensive evaluations of the respective approach. Each paper, in turn, comes with an evaluated artifact that provides the option to reproduce the presented results. In summary, Phasar has become a mature analysis framework that is evaluated, recognized and adopted by research and industry.

With ReproDroid we contributed an open source framework that allows anyone to evaluate analyses automatically and in a reproducible fashion on given benchmarks. Consequently, ReproDroid simplifies the benchmarking process, which was often performed manually before. Therefore and since evaluations such as benchmark executions are indispensable to show the effectiveness and efficiency of analyses, ReproDroid was not only used by us in our five subsequent publications to drive the associated evaluations but also by others. This versatile usage of ReproDroid best shows its impact in the community. In future, it could even become more important as a driver for competitions in the area of Android taint analysis, for example. Please note that each of our publications involving ReproDroid comes with an evaluated artifact and/or an open source repository. The related repositories in sum acquired 65 stars. All frameworks, tools and benchmarks released are also available on the respective website of the CRC.[15]

In summary, Subproject B4 has left its mark in the area of proof-carrying services or, in general, on soft- and hardware verification and analysis. Due to the publications made as well as the implementations and artifacts contributed, this mark has become persistent such that future researchers and practitioners can take our ideas, understand our results, use our tools and frameworks, and continue what has been started.

Bibliography

[Bey22]   Beyer, D.: Progress on Software Verification: SV-COMP 2022. In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS*

---

[14]Github repository: `https://github.com/secure-software-engineering/phasar` (04/24/2023)

[15]`https://sfb901.uni-paderborn.de/projects/tools-and-demonstration-systems`

                  2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part II. Ed. by FISMAN, D.; ROSU, G. Vol. 13244. Lecture Notes in Computer Science. Springer, 2022, pp. 375–402.

[BK11]   BEYER, D.; KEREMOGLU, M. E.: CPAchecker: A Tool for Configurable Software Verification. In: *CAV*. Ed. by GOPALAKRISHNAN, G.; QADEER, S. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 184–190.

[DKP09]   DRZEVITZKY, S.; KASTENS, U.; PLATZNER, M.: Proof-carrying Hardware: Towards Runtime Verification of Reconfigurable Modules. In: *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2009

[DKP10]   DRZEVITZKY, S.; KASTENS, U.; PLATZNER, M.: Proof-Carrying Hardware: Concept and Prototype Tool Flow for Online Verification. In: *International Journal of Reconfigurable Computing* 2010 (2010)

[HSKL08]   HUFFMIRE, T.; SHERWOOD, T.; KASTNER, R.; LEVIN, T.: Enforcing memory policy specifications in reconfigurable hardware. In: *Computers & Security* 27 (2008), no. 5–6, pp. 197–215.

[JPWW14] JAKOBS, M.; PLATZNER, M.; WEHRHEIM, H.; WIERSEMA, T.: Integrating Software and Hardware Verification. In: *IFM*. Ed. by ALBERT, E.; SEKERINSKI, E. Vol. 8739. Lecture Notes in Computer Science. Springer, 2014, pp. 307–322.

[JW14]   JAKOBS, M.; WEHRHEIM, H.: Certification for configurable program analysis. In: *SPIN*. Ed. by RUNGTA, N.; TKACHUK, O. ACM, 2014, pp. 30–39.

[JW15]   JAKOBS, M.; WEHRHEIM, H.: Programs from proofs of predicated dataflow analyses. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*. Ed. by WAINWRIGHT, R. L.; CORCHADO, J. M.; BECHINI, A.; HONG, J. ACM, 2015, pp. 1729–1736.

[JW17]   JAKOBS, M.; WEHRHEIM, H.: Programs from Proofs: A Framework for the Safe Execution of Untrusted Software. In: *ACM Trans. Program. Lang. Syst.* 39 (2017), no. 2, 7:1–7:56.

[LJM12]   LOVE, E.; JIN, Y.; MAKRIS, Y.: Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition. In: *IEEE Transactions on Information Forensics and Security* 7 (1 Feb. 2012), no. 1, pp. 25–40

[LP09]   LÜBBERS, E.; PLATZNER, M.: ReconOS: Multithreaded Programming for Reconfigurable Computers. In: *ACM Transactions on Embedded Computing Systems (TECS)* 9 (1 Oct. 2009), 8:1–8:33

[LPP+22]   LUO, L.; PAUCK, F.; PISKACHEV, G.; BENZ, M.; PASHCHENKO, I.; MORY, M.; BODDEN, E.; HERMANN, B.; MASSACCI, F.: TaintBench: Automatic real-world malware benchmarking of Android taint analyses. In: *Empir. Softw. Eng.* 27 (2022), no. 1, p. 16.

[PBW18]   PAUCK, F.; BODDEN, E.; WEHRHEIM, H.: Do Android taint analysis tools keep their promises? In: *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*. Ed. by LEAVENS, G. T.; GARCIA, A.; PASAREANU, C. S. ACM, 2018, pp. 331–341.

[PW19]   PAUCK, F.; WEHRHEIM, H.: Together strong: cooperative Android app analysis. In: *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*. Ed. by DUMAS, M.; PFAHL, D.; APEL, S.; RUSSO, A. ACM, 2019, pp. 374–384.

[SGP+22]   SCHUBERT, P. D.; GAZZILLO, P.; PATTERSON, Z.; BRAHA, J.; SCHIEBEL, F.; HERMANN, B.; WEI, S.; BODDEN, E.: Static data-flow analysis for software product lines in C. In: *Automated Software Engineering* 29 (Mar. 2022), no. 1, p. 35.

[SHB19]   SCHUBERT, P. D.; HERMANN, B.; BODDEN, E.: PhASAR: An Inter-procedural Static Analysis
           Framework for C/C++. In: *Tools and Algorithms for the Construction and Analysis of
           Systems*. Ed. by VOJNAR, T.; ZHANG, L. Cham: Springer International Publishing, 2019,
           pp. 393–410

[SHB21]   SCHUBERT, P. D.; HERMANN, B.; BODDEN, E.: Lossless, Persisted Summarization of Static
           Callgraph, Points-To and Data-Flow Analysis. In: *35th European Conference on Object-
           Oriented Programming (ECOOP 2021)*. Ed. by MØLLER, A.; SRIDHARAN, M. Vol. 194.
           Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss
           Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 2:1–2:31.

[TW18]   TÖWS, M.; WEHRHEIM, H.: Information Flow Certificates. In: *ICTAC*. Ed. by FISCHER, B.;
           UUSTALU, T. Vol. 11187. Lecture Notes in Computer Science. Springer, 2018, pp. 435–454.

[WDP14]   WIERSEMA, T.; DRZEVITZKY, S.; PLATZNER, M.: Memory Security in Reconfigurable Comput-
           ers: Combining Formal Verification with Monitoring. In: *Proceedings of the International
           Conference on Field-Programmable Technology (FPT)*. 2014, pp. 167–174

[WSW13]   WONISCH, D.; SCHREMMER, A.; WEHRHEIM, H.: Programs from Proofs - A PCC Alternative.
           In: *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Peters-
           burg, Russia, July 13-19, 2013. Proceedings*. Ed. by SHARYGINA, N.; VEITH, H. Vol. 8044.
           Lecture Notes in Computer Science. Springer, 2013, pp. 912–927.

Subproject C1:
Robustness and Security

Johannes Blömer[1], Fabian Eidens[1], Tibor Jager[2], David Niehues[2],
Christian Scheideler[1]

1 Department of Computer Science, Paderborn University,
   Paderborn, Germany
2 School of Electrical, Information and Media
   Engineering, University of Wuppertal, Wuppertal,
   Germany

1   Introduction

In Subproject C1 of the CRC, we developed methods and techniques that ensure high robustness and security in OTF markets, taking into account the specific characteristics and demands of such a market. Although robustness and security share the high level objectives of availability and acceptance, in practice, they require quite different techniques. Robustness usually (but not exclusively, think about denial-of-service attacks) deals with unforeseen but not necessarily malicious behavior of participants, whereas malicious attacks are at the very core of security. For a service to be secure, it needs to satisfy several key properties. First, there need to be mechanisms that guarantee the integrity of the communication and the authenticity of communication partners. Second, in many cases communication needs to be confidential. Third, privacy-preserving mechanisms are required to protect the identity of communication parties, i.e., to guarantee the anonymity of parties, and to protect sensitive data like business secrets or customer data. While these are the standard requirements and properties from security and privacy, and several techniques exist to realize these goals, even simultaneously, online markets such as the OTF market have special characteristics that necessitate specialized and enhanced methods. For example, the decentralized nature of OTF markets mostly prohibits the use of centralized techniques for access control. Furthermore, in some applications, e.g., machine learning based applications, the quality of services offered by service and software providers may depend heavily on the quality of (sensitive) data provided by customers, such as learning data. Hence there is a need to make sensitive data available for learning algorithms while still preserving their confidentiality and privacy.

For a service to be robust, it should be available 24/7 and withstand faults as well as Byzantine behavior as much as this is possible. This is particularly important for an open OTF market, where many different stakeholders need to interact without any prior trust relationships and customers are only willing to use it if its services are highly available. A

bloemer@upb.de (Johannes Blömer), fabian.eidens@uni-paderborn.de (Fabian Eidens), tibor.jager@uni-wuppertal.de (Tibor Jager), niehues@uni-wuppertal.de (David Niehues), scheideler@upb.de (Christian Scheideler)

critical aspect of such a market is, for example, any kind of information needed for the composition of software solutions, such as reputation data. Ideally, a solution for such information systems should even withstand insider attacks since the code used for the organization of the OTF market might be freely available to everyone. Another important aspect is an appropriate infrastructure interconnecting the market participants that has a low maintenance overhead and that can handle even large churn (i.e., a high arrival and departure rate of market participants) without seriously affecting the exchange of information.

In the area of robustness, we first focused on robust information systems. A huge problem for robust information systems are denial-of-service (DoS) attacks. There are basically two approaches to counter DoS attacks: stopping DoS attacks, for example, by filtering them, or setting up a system that remains available despite a DoS attack. Stopping DoS attacks is usually a hard problem as it requires interactions with Internet service providers or security agencies, so we focused on systems that remain available despite DoS attacks. Various such systems have already been proposed when DoS attacks are initiated by outsiders, i.e., attackers that do not know the setup of the system. We, instead, considered attacks by insiders, i.e., attackers that know everything about the system and can use that information in order to start DoS attacks on a limited number of its servers in order to make certain parts of the information unavailable to legitimate requests. Our findings are summarized in Section 2.4. Later, we also focused on the problem of protecting an overlay network against DoS attacks. For an overlay network to be scalable, its degree should be at most polylogarithmic in the number of nodes, since a high degree also means a high maintenance overhead. However, the lower the degree, the easier it is to start so-called Eclipse attacks, i.e., attacks that isolate parts of the network from the rest. These Eclipse attacks can, for example, be performed by starting a DoS attack on the neighborhood of a targeted node so that it cannot interact anymore with the rest of the network. Our approach to defend against such kinds of attacks is to continuously change the topology of the network, and to do this so quickly that an attacker cannot keep up with the changes. Our findings are summarized in Section 2.5.

In security, we focused on techniques that guarantee strong authentication of data and entities while also preserving user privacy. To achieve this we construct enhanced signature schemes and anonymous credentials. Finally, to help actors in OTF markets find appropriate services we study secure and anonymous reputation systems. We begin by briefly discussing reputation systems and their use in the context of OTF. In OTF markets, reputation systems can complement certificates and provide important information about the quality and trustworthiness of service or software providers. However, reputation systems as currently used in many online markets face several security and privacy issues. In particular, to avoid retribution for negative ratings, anonymity of ratings seems to be a desirable property. Anonymity itself, on the other hand, creates problems, such as skewing the reputation score of a service by a flood a negative ratings. To overcome these problems, we have identified key properties of secure and privacy-preserving reputation systems. These properties have been summarized into precise definitions of cryptographic reputation systems that did not exist prior to our work. We also provide efficient constructions of reputation systems. These results are described in more detail in Section 2.1. To authenticate data and users, one can use signatures and credentials, respectively. However, in the OTF market we often need enhanced versions of signatures or credentials to meet

the requirements of the markets. To meet these demands, we defined and constructed several novel cryptographic principles. These are described in more details in Section 2.2 (updatable anonymous credentials) and Section 2.3 (verifiable random functions). Whereas digital signatures authenticate data, credentials authenticate entities, usually based on attributes of entities. If credentials are used across many applications, they allow for tracking and may reveal private information of entities, e.g., in the OTF markets, entities may be users that contact many service providers and, given a composed service, have to use different compute centers. Hence anonymity is a concern. This is provided by anonymous credentials, a cryptographic technique developed in the last 20 years. In our research we realized efficient constructions for anonymous credentials with additional features, e.g., updatability. As an application of the credentials, in transfer project T2 of the CRC 901 we design privacy-preserving incentive systems (see page 237). Verifiable random functions are enhanced digital signatures whose additional properties make them attractive and useful for applications in the OTF market like consensus systems and public-key distribution systems. Usually, constructions of verifiable random functions rely on the so-called random oracle model. However, it is well-known that random oracles cannot be realized and need to be replaced by standard hash functions, leading to constructions whose security is only heuristic. We developed new verifiable random functions that do not rely on the random oracle model and are more efficient then previous constructions. The techniques introduced in this work also have applications beyond verifiable random functions.

2 Main Contributions

In this section, we review following five highlights from the research in Subproject C1:

- cryptographic reputation systems (Section 2.1)

- updatable anonymous credentials (Section 2.2)

- efficient verifiable random functions without random oracles (Section 2.3)

- insider-resistant distributed storage systems (Section 2.4)

- construction and maintenance of robust overlays (Section 2.5).

2.1 Cryptographic Reputation Systems

In an OTF market, users will contact OTF providers who, in turn, will contact service and/or software providers. In some cases, these contacts will not be based on substantial prior direct experience. In particular, OTF service providers rely on software providers that they use only occasionally. In these cases, as in others, it is valuable to have information about service and software providers that helps customers (users or service providers) to assess the quality and trustworthiness of other providers. For software and service providers, certificates may play an important role. However, the dynamics of an OTF market reduces the availability of trustworthy certificates compared to more traditional markets. Moreover, certificates may not say anything about the quality of (recent) products

and services offered by certified providers. As an alternative, reputation systems become more important, as witnessed for decades in online consumer markets.

Reputation systems provide valuable information about previous transactions and are popular tools to measure the trustworthiness of interacting parties. This measurement relies on the existence of a large number of ratings for one specific service or product. However, in most practical applications the process of rating reveals much information about the rater, besides the actual rating. Providers of reputation systems may use this information in many different ways that are not necessarily desired by the users, such as profiling users. Moreover, users can feel compelled to rate "dishonestly/benevolently" when they fear negative consequences from negative ratings. Therefore, it is important that raters at least have the choice to not reveal more information than the actual rating. Besides that, reputation systems need to be protected against various attacks to provide trustworthy, reliable, and honest ratings. These attacks include self-rating attacks (also known as self-promoting attacks), Sybil attacks, whitewashing attacks, bad mouthing attacks, ballot stuffing attacks, and value imbalance attacks. Both the privacy concerns and the prevention of attacks are discussed frequently in the literature, e.g., [BPS+17; ZWC+16], albeit they were often not considered simultaneously. Further important security properties for reputation systems are anonymity, (public) linkability, traceability, and non-frameability, as discussed in e.g., [BJK15; ZWC+16]. Anonymity means that ratings of honest users are indistinguishable, whereas public linkability requires that anyone can decide whether or not two ratings for the same product were created by the same user. Also, ratings need to be traceable: The identity of any rater can be determined by a designated system manager. In the course of this, non-frameability guarantees that honest parties are not blamed of having rated some product, when they did not. The combination of traceability and non-frameability enables penalizing dishonest behavior.

Two different approaches to define and prove the security of cryptographic primitives are used in the literature: experiment based and simulation-based. In experiment-based security, so-called security experiments and games are defined mathematically, in which an adversary plays against a challenger running a cryptographic primitive. In a security proof for a cryptographic primitive one then shows that no efficient algorithm or adversary can win this game, except with tiny probability of success. Simulation-based security definitions define security properties by describing ideal scenarios that make use of, practically not realizable, trusted parties. A security proof for a cryptographic primitive shows that an adversary trying to attack the primitive does not have a significantly better chance of succeeding than an adversary in the idealized scenario. Experiment-based security definitions and proofs tend to be more efficient and easier to understand. In comparison, simulation-based security usually offers better security guarantees, in particular if cryptographic primitives are combined or composed with each other. The important simulation-based framework for proving the security of composed cryptographic primitives is Ran Cannetti's *universal composability (UC)* framework [Can01]. In our research we considered both, experiment-based and simulation-based, definitions and constructions.

Experiment-Based Security of Reputation Systems

In [BJK15] we gave a first definition of a cryptographically secure and anonymous reputation systems. We also provided a construction of such a system based on group signature schemes. Group signatures are one of the most important primitives for privacy-preserving

cryptography. A group signature allows a group of users to sign documents on behalf of the group without revealing the identity of the specific member. In our system, we establish a separate group for each product or service, consisting of all users that bought the product, and hence have the right to rate the product. The reputation system provides anonymity, traceability, strong-exculpability, verifier-local revocation, and public linkability. Except for revocation, these properties have already been discussed above. Revocation mechanisms are a security check against the misuse of anonymity (or better pseudonymity), since it allows a system manager to revoke the rights of users, i.e., in group signatures the right to sign messages and in reputation systems the right to publish the rating. A system has verifier-local revocation, if revocation messages only have to be sent to signature or rating verifiers but not to individual signers or raters. It is well known how to realize anonymity, traceability, non-frameability, and revocation in the context of group signatures, although not necessarily simultaneously and for many groups in parallel. Our construction of a reputation system achieves this, and adds some properties specific to reputation systems. The construction is based on a group signature scheme by Boneh, Boyen, and Shacham [BBS04] and the dynamic version of the scheme presented by Delerablée and Pointcheval [PS16]. These schemes already give us anonymity, traceability, and strong-exculpability. To achieve verifier-local revocation we modify a technique by Nakanishi and Funabiki [NF06]. With the same technique we achieve public linkability. Note that anonymity of group signatures does not imply anonymity in our reputation system. This is due to the fact that providers control the groups corresponding to several products. Hence, they may combine information for different groups to violate anonymity. To prevent this, in our construction we employ a system manager that contributes a trustworthy component to each group public key.

UC-Secure Reputation Systems

Typically, reputation systems are used in combination with other applications. Common experiment-based security definitions, such as the ones defined and used in [BJK15], provide next to no security in such circumstances. With the universal composability framework (UC) of R. Canetti [Can01] there exists a methodology that guarantees security even in composed applications. Informally, in UC the execution of a real-life protocol is compared to the execution of an ideal protocol. If the real-life and ideal protocol executions are indistinguishable, then the real-life protocol is UC-secure. Based on this security definition, Canetti formulates a composition theorem that states that any UC-secure protocol is also secure when it is composed with other protocols.

In [BEJ18], first we present an ideal functionality for reputation systems in the UC framework. Our ideal functionality prevents all previously mentioned attacks and provides anonymity, public linkability, traceability, and non-frameability. Based on this, and extending the construction in [BJK15], we construct a reputation system where (a) users can rate each other's products, (b) there is no separation of customers and providers, (c) security is preserved under composition, i.e., we present an efficient protocol for reputation systems that realizes the ideal functionality for reputation systems. All three properties are highly relevant for the use of reputation systems in an OTF market. Here, reputation systems are embedded into a larger (security) system. Hence, universal composability is required. Moreover, participants in OTF markets may simultaneously or at different times play the roles of a user and of a service or software provider. Hence, it is mandatory,

that reputation systems satisfy properties (a) and (b), as well. On a technical level, our reputation system is influenced by techniques known from Σ-protocols and (dynamic) group signatures, similarly to the scheme in [BJK15]. However, to achieve UC-security we need to employ numerous advanced techniques known from other constructions of UC-secure cryptographic primitives. Somewhat surprisingly, the resulting system, in addition to being UC-secure, is also more efficient and more flexible than the scheme in [BJK15].

2.2 Updatable Anonymous Credentials

In current systems, classical authentication of a user at a provider usually involves that a user provides identifying information (e.g., name, email) combined with some user specific secret (e.g., passport, password). Presented with this, the provider grants the user access to its service. Furthermore, while the user is interacting with the provided service, the provider stores additional data of the user in a database. For example, profile information such as address, day of birth, and user's preferences. Already this simple example, is a threat to users' privacy, since they have no sovereignty over their data and data becomes stores and associated with their identity with no or limited benefit for the user. This becomes even more serious if we consider several providers that pool their databases (for example, after an acquisition), which allows them to link users across services.

Anonymous credentials employed in such a scenario enables anonymous authentication of users at providers such that no identifying information is revealed. For this, an anonymous credential encodes information about the user in certified attributes and providers can define (access) policies over attributes. Therefore, an authentication via an anonymous credential only proves that the user in question has a credential on certified attributes that satisfy the policy. For example, the policy checks if the user has a valid subscription for the service of the provider. With anonymous credentials, the provider then only learns that the end date of the subscription is before or after the current date. In general, a policy can be interpreted as a statement and through authentication a provider only learns the truthfulness of the statement based on the attributes of a user. Hence, authentication via anonymous credentials is more expressive than classical authentication and leaks minimal information. In the literature, this extension to anonymous credentials [CL01; CL04; PS16] is referred to as attribute-based anonymous credentials and many more extensions are given in the literature, e.g., delegation of credentials [BCC+09; BB18; CL19; CDD17; MSBM22], revocation [CL01; CL02; CKS10], auditing [CLNR14], and expressive policies [CG08; BBB+18].

Beyond that, anonymous credentials solve the problem that providers operate databases that includes user data that, from a privacy-preserving perspective, are better suited to be stored on the user side. Intuitively, instead of storing user information in a database row, one can employ an anonymous credential system. Then, the row is encoded as an attribute vector and certified by an anonymous credential which is then stored on a user's device. In this setting, users get their attributes certified by a provider acting as an issuer of anonymous credentials. However, the question arises, how providers and users can update their attributes as it is a common process in a system that uses a database. There, the provider would just update some entries in a row associated to a user. For this,

we introduced updatable anonymous credentials (UAC) in [BBDE19] allowing privacy-preserving updates of attributes certified a credential. To do this, the user has to contact the original issuer of the credential and both agree on an update that they want to execute. The result of this is a new credential on update attributes, where the update process does not leak the attributes or the credential to the issuer. The issuer only learns which update was performed, i.e., if the update was a "+7" on a point counter attribute the issuer only learns that it updated some point counter with "+7". With this short example in place, let us describe the roles and processes of an UAC system before we show how we can instantiate an UAC system.

Roles and Protocols

In an UAC system there are users, issuers, and verifiers. Users can get their attributes certified in a credential by an issuer. Hence, issuers are responsible for generating credentials on attributes and verifiers check the validity of credentials with respect to a policy.

To describe the protocols that the different parties (roles) execute, let us expand our subscription example. Here, a user wants to get a credential from an issuer certifying that the user has a valid subscription. For this first issuance of a credential the user shows a recipe of the subscription to validate his assertion of a valid subscription. This can also be realized with an anonymous payment. However, this is outside of the system and for the example we just assume that the issuer can be sure that the user has a valid subscription. Following this, the user and issuer execute a so-called issue protocol. Here, the issuer generates a credential on the subscription end date by encoding it in an attribute called `sub_end`. Additionally, the issuer adds a second attribute to the credential, called `actions`, which is initialized to be 0. The attribute `actions = 0` is given to any user that joins the system. Next, using the issued credential, the user can authenticate to a verifier via a show protocol to get access to the subscribed service as described above. If the user now performs some predefined actions, e.g., by using a specific feature of the service, the issuer offers the user to update its credentials, i.e., it increments the `actions` attribute by 1. For this, the user and issuer agree on the update (in the form of an update function) and execute an update protocol. The result is a new credential for the user on attributes `sub_end` (unchanged) and `actions = 1`. Furthermore, a verifier can give a discount on other services or features of the service if the user has a valid subscription and has performed more than 50 actions. For this, the user and verifier execute a show protocol in which the user proves that it has a valid subscription and now also proves that its `actions` attribute is greater than 50. This show protocol does not leak any information about the actual `actions` count or the end date of the subscription. In general, the practical features of UAC combined with privacy-preserving protocols seem as though they require heavy cryptographic techniques that are inefficient in real-world applications. The contrary is the case and was shown by a formal analysis of the UAC system and a prototype implementation in [BBDE19; BEHF21].

Efficient Instantiation

In the following, we present how UAC can be efficiently instantiated with modern cryptographic building blocks in which efficient implementations are available. Up to now, we referred to the attributes as certified by a credential. Concretely, in UAC a credential is a

digital signature on a message vector that represents an attribute vector. That means an issuer generates its own public-secret key pair of a digital signature scheme under which it issues credentials. Furthermore, the proofs that the user has to generate, e.g., in a show protocol, are done via a proof system called (non-interactive) zero-knowledge arguments of knowledge. Zero-knowledge arguments of knowledge are systems that generate an efficiently verifiable proof string with two security properties: zero-knowledge and argument of knowledge. Zero-knowledge means that the proof does not leak any information about the secrets of the proof (also called witness), such as the attributes. Argument of knowledge guarantees that no adversarial user can convince a verifier, i.e., generate a valid proof, without having a valid witness. This means if the UAC policy to be proven cannot be satisfied by user's attributes, no adversarial user can generate a valid proof. Hence, the security properties of the proof system protects the privacy of the user and the interests of the verifier.

With these building blocks, in place we can start describing the technical details of issue and update protocols. Since an issue protocol is just a special case of an update protocol we just describe the latter. To get a credential on updated attributes, suppose a user with an existing credential on attribute vector containing `sub_end` and `actions` attributes agreed on an update function that updates the `actions` attribute by "+7". The user prepares this update by sending the issuer a commitment on the updated attribute vector. This commitment does not leak any information about its content and guarantees that the user cannot change the committed values later in the protocol. In case of an issue protocol, the update function just sets the attributes to be issued to its starting values. Next, the user computes a proof that shows that the commitment is correctly formed. This means that it contains attributes of a valid credential and the update was correctly prepared. Then, the issuer checks the proof and, if it is valid, it digitally signs the commitment and sends the result back to the user as his credentials.

2.3 Efficient Verifiable Random Functions without Random Oracles

We developed more efficient constructions of so-called *variable random functions* (VRFs), which can be seen as enhanced digital signature schemes with additional properties. Verifiable random functions play an important role in several applications relevant to Subproject C1 of the CRC-901. Specifically, VRFs are a core building block of the family modern consensus mechanisms called *proof-of-stake*, which are part of AP 1 of Subproject C1. Moreover, VRFs are used in constructions of verifiable distributed public-key distribution systems, such as CONIKS [MBB+15]. Such verifiable distributed public-key distributions systems are relevant to the decentralization of the components for On-the-Fly Computing described in AP 3.2.

The Random Oracle Model

In practical modern cryptography, the so-called *random oracle model* (ROM) introduced by Bellare and Rogaway is often used. In this model, one or several hash functions are modeled as so-called *random oracles* (ROs). A RO can be queried on specific inputs from the domain of the hash functions by all parties that are active in the context of the cryptographic scheme. Each random oracle maintains an initial list that maps hash function

inputs to outputs. Every time the RO is queried on an input x, it checks whether its list contains an entry for x. If this is not the case, it draws an element y from the range of hash functions uniformly at random and stores the mapping $x \mapsto y$ in its list and returns y. If such a mapping $x \mapsto y$ already exists in the list, y is retrieved and returned. Moreover, it is common to allow *programming* the RO in the proof of the security of a cryptographic scheme. That is, inside the proof specific mappings of inputs and outputs may be chosen as long as the distribution of the outputs remains provably indistinguishable from the distribution of a non-programmed oracle. Note that this is a very strong idealization of a cryptographic hash function, which provides not only all standard security properties such as collision resistance or (second) preimage resistance but also many further very strong properties beyond this, such as *programmability* in a security proof, which is not possible for a fixed concrete function such as SHA-3. Unfortunately, it is known that random oracles can not be instantiated in general [CGH04]. Therefore, a concrete practical instantiation of a construction that is only proven secure in the ROM only achieves heuristic security. As a result, from a practical perspective, it would be preferable to have efficient cryptographic schemes that can be proven secure outside the ROM. Similarly, from a theoretical perspective, constructing such cryptographic schemes helps advancing our understanding of what can be achieved outside the ROM and when the ROM is inherently necessary.

New Techniques for Verifiable Random Functions and Further Applications

VRFs are a public-key primitive, where a public *verification key* vk identifies a function $F_{vk} : X \to Y$ for some domain X and some range Y. However, vk does not allow to efficiently evaluate F_{vk}. The respective *secret key* sk then allows the following two functionalities:

1. Evaluating F_{vk} on any input $x \in X$ and thus obtaining $y = F_{vk}(x)$ in an efficient way.

2. Generating a proof of correct evaluation π that can be efficiently verified with the help of vk. That is, vk and π together allow to verify that $y = F_{vk}(x)$ holds without the need to know sk.

Finally, we require that even for an adversarially chosen input $x \in X$ it remains impossible for any efficient algorithm to distinguish $y = F_{vk}(x)$ from a $y' \in Y$ that is chosen uniformly at random if π is not known.

We developed new verifiable random functions that do not rely on the ROM and are significantly more efficient than previously known constructions in terms of the size of vk, sk and π [JN19; JKN21]. The techniques also turned out to have further applications beyond VRFs. *Identity-Based Encryption* (IBE) is a type of public-key encryption where, there is only a single *master public-key* mpk known to all parties and a respective *master secret-key* msk only known to a trusted third party. Using msk, the trusted third party can then issue *user secret keys* sk for arbitrary identities, e.g., email addresses, and provide the respective users with them. It then suffices to know mpk and a user's identity to encrypt a message for the user such that only that user can decrypt the ciphertext efficiently. Based on the same techniques that we applied in the context of VRFs, we also developed more efficient IBE schemes. In [JKN21], we describe new more efficient IBE schemes with security under assumptions related to the hardness of the discrete logarithm problem in elliptic curve groups and under the learning with errors (LWE) problem, which provides

post-quantum security. Moreover, in [Nie21a] more efficient constructions of IBEs are described.

Verifiable Random Functions with Optimal Tightness

Another aspect of efficiency of cryptographic schemes is called *tightness*. Security in modern cryptography is often proven by reducing the security of the cryptographic scheme to the intractability of some computational problem, such as the discrete logarithm problem, the factorization problem, or the learning with errors problem. However, the *quality* of such reductions can vary in the tightness with which they relate the security of the cryptographic scheme to the intractability of the respective computational problem. That is, applying the reduction to an algorithm $\mathcal{A}$ running in time $t_{\mathcal{A}}$ that breaks the security of a cryptographic scheme with probability $\epsilon_{\mathcal{A}}$ yields an algorithm $\mathcal{B}$ that solves the computational problem in time $t_{\mathcal{B}} \geq t_{\mathcal{A}}$ with probability $\epsilon_{\mathcal{B}} \leq \epsilon_{\mathcal{B}}$. In order to achieve efficient cryptographic schemes, we want to construct reductions that have a so-called *loss* $\ell := (t_{\mathcal{B}}/\epsilon_{\mathcal{B}})/(t_{\mathcal{A}}/\epsilon_{\mathcal{A}})$ that is as small as possible and ideally a small constant. Reductions with a small loss have the advantage that strong security guarantees can be achieved while relying on smaller instances of the computational problem. For example, a tight reduction could allow us to use groups of smaller size when relying on the intractability of the discrete logarithm problem, which would yield smaller keys and make algorithms more efficient. This approach is also known as *concrete security* and thoroughly discussed in [BR09]. This raises the natural question of how tight reductions for cryptographic schemes can be. We advanced the state of the art in the research area by providing the first lower bound for the loss of reductions from the security of VRFs to non-interactive hardness assumptions and providing the first construction of a VRF with an accompanying security proof that meets this bound [Nie21b].

2.4   Insider-Resistant Distributed Storage Systems

In recent years, the use of online services has increased significantly. For instance, communicating with friends via social media platforms, sharing videos via YouTube, or shopping online via Amazon. This induces the necessity to store large amounts of data online in such a way that they can be managed and accessed efficiently. Distributed storage systems constitute one of the most natural approaches for the implementation of such a storage. Popular examples include storage solutions offered by Google, Apple, and Amazon. We considered distributed storage systems that are defined as a network consisting of several servers that provide a lookup and update operation. If only a lookup operation but no update operation is provided, we call the system a *distributed information system*.

Since availability and retrievability of the stored data is a key aspect of distributed storage systems, these systems should have various mechanisms in place to protect them against adversarial attacks. One of the biggest threats distributed storage systems are exposed to are crash failures. A server that experiences a crash failure is not available anymore, meaning that it neither responds to any requests nor performs any further operations. Crash failures can be temporary or permanent and can have many causes, such as maintenance work, hardware or software failures, or DoS attacks. Especially crash failures caused by DoS attacks can pose a serious threat, since they usually are unpredictable, hard to prevent,

and can cause the unavailability of a server for some time. Besides crash failures, storage failures also constitute a big threat to distributed storage systems. A server that experiences a storage failure may hold arbitrarily corrupted data in its storage without being aware of that.

While a crash failure can easily be detected using crash failure detectors, this does not hold for massive storage failures. Instead, the distributed storage system needs to implement techniques and methods in order to work correctly despite the existence of servers with storage failures. Storage failures may not only be caused by malicious adversaries, they may also occur due to technical errors, such as disk faults or physical interconnect failures. For instance, in 2008 Amazon's S3 storage service experienced a multi-hour downtime due to a single bit corruption resulting in monetary loss for Amazon and the unavailability of data stored at the S3 storage service.

The predominant approach in distributed storage systems to deal with the threat of failures is to use redundancy and information hiding: The idea behind this is that information that is not only stored at a single server but also replicated on multiple servers is more likely to remain accessible during an attack, in particular if the adversary does not know the storage locations of the redundant data items. For example, if a logarithmic number of copies of each data item is distributed among the servers in the system, and the adversary is not aware of these locations, then it is easy to see that with high probability a copy of each data item is still accessible if the adversary crashes less than half of the servers. However, the situation is completely different when considering an insider adversary, i.e., someone who has complete knowledge of the system and may use this knowledge to crash a large fraction of the servers. Since information cannot be hidden anymore in this case, it seems unavoidable to replicate each data item across more than t servers in order to remain accessible if the system is under an attack that crashes t servers. Unfortunately, in this case the storage overhead becomes very large when considering adversaries that may crash a large fraction of the servers. However, it turns out that this dilemma can be circumvented when using coding, which is one of the key ideas we used in the development of robust storage systems.

Concretely, our goal was to develop distributed information and storage systems that provide efficient lookup and write protocols that work provably correctly despite the existence of an insider adversary that may attack a large fraction of the servers by causing crash failures or a special type of storage failure. In this context, by *efficient* we mean at most polylogarithmic in the number of servers, and by a *large fraction* of attacked servers we mean asymptotically much larger than polylogarithmic, such as $O(n^{1/\log\log n})$, with n being the number of servers, or even up to a constant fraction of all servers. At the same time, we ensure the additional amount of storage required by each server to be limited by at most a logarithmic factor.

Basic IRIS

Our first result was IRIS, a distributed information system that is provably robust against an insider adversary that crashes up to $O(n^{1/\log\log n})$ servers while requiring only a constant storage redundancy. The main innovation in this system is the development of a technique for the efficient encoding of the data items stored in the system with each other using a hierarchical coding strategy that is based on the structure of a k-ary butterfly ($k = \Theta(\log n)$)

and a simple parity-based code. This technique allows to specify a lookup protocol that guarantees to correctly serve each lookup request for any data item with polylogarithmic work at each server and polylogarithmic time, although the adversary may crash up to $O(n^{1/\log\log n})$ servers.

Enhanced IRIS

We then extended IRIS to Enhanced IRIS, which is able to tolerate even up to a constant fraction of all servers to be crashed. Except for the storage redundancy, which increases to a logarithmic factor, Enhanced IRIS still guarantees the same properties as Basic IRIS. The main idea behind this extension of Basic IRIS is to not only use a k-ary butterfly as the underlying topology for the encoding, but to additionally make use of permutations that fulfill certain expansion properties in order to spread the encoding information even further among the servers. IRIS and Enhanced IRIS were presented in [ES15].

RoBuSt

While Basic IRIS and Enhanced IRIS are distributed information systems that provide only a lookup functionality, we later developed RoBuSt, a distributed storage system that provides both lookup and update functionality [ESS14]. More precisely, RoBuSt is a distributed storage system that correctly handles lookup and write requests in polylogarithmic time and with polylogarithmic work despite the existence of an insider adversary that crashes up to $O(n^{1/\log\log n})$ servers. On top of that, RoBuSt requires only a logarithmic storage redundancy. RoBuSt reuses the k-ary butterfly encoding approach introduced with Basic IRIS with the additional ingredient of a clever arrangement of the data items stored in the system into so-called buckets and an appropriate strategy for traversing the buckets efficiently.

OSIRIS

We further strengthened the adversary considered in such a way that it now may not only crash servers, but instead even corrupt the storage of up to $O(n^{1/\log\log n})$ servers. Here, we confined ourselves to the corruption of the data stored at the servers while assuming the protocols and main memory of the servers to be reliable. This kind of attack can also be interpreted as a DNS spoofing attack. The main challenge in this setting is that, in contrast to crashed servers, there is no way to efficiently detect corrupted servers. Hence, we needed to add techniques for verifying the validity of data. By appropriately interweaving techniques from the field of authenticated data structures, namely Merkle trees, with techniques developed for IRIS and RoBuSt, we developed OSIRIS, a distributed storage system that is provably robust against an insider adversary that may corrupt the storage of up to $O(n^{1/\log\log n})$ servers. At the same time, OSIRIS correctly answers any set of lookup and update requests in polylogarithmic time and with polylogarithmic work per server while requiring a logarithmic redundancy only [Eik16].

2.5   Construction and Maintenance of Robust Overlays

A key design goal for our OTF market infrastructure is to be open and permissionless. We desire this for two reasons. On the one hand, we want to put little to no boundaries on new parties entering the market to keep it competitive. In particular, established market participants should not be able to prevent new competitors from entering. On the other hand, any participant should be able to leave the market without affecting the functionality, i.e., the network should not be constructed *around* one powerful party that handles a significant amount of market transactions. Therefore, the OTF market infrastructure lends itself to be implemented in a peer-to-peer (P2P) fashion. The P2P approach has proven to be a useful technique for constructing resilient, decentralized systems. In a P2P architecture, the participants (which we will call nodes in the remainder) are connected via the Internet and form a logical network topology, also known as an overlay network or simply overlay. Within the overlay, each node has a logical address and logical links that allow it to search and store information in the network. Ideally, the topology has no single point of failure, so nodes can leave the network without disrupting the functionality. Furthermore, it is designed to scale with any number of nodes.

A fundamental requirement for all applications built on P2P networks is reliable communication between the nodes, i.e., each node should be able to send a message to another node at all times. Of course, this also holds for our OTF market infrastructure. Ensuring reliable communication is complicated by the fact that in every large-scale system, errors and attacks are the rule rather than the exception. Together with the fact that nodes may frequently leave or enter the system on their own accord, this implies a massive amount of so-called *churn*, i.e., changes in the set of nodes. Therefore, we investigated robust distributed protocols that maintain connected overlays despite heavy churn.

Throughout our work, we used the de-facto standard model for P2P algorithms. We assume that time proceeds in synchronous[16] rounds and observe a dynamic set of nodes $\mathcal{V} := (V_0, V_1, \dots)$ such that V_t is the set of nodes in round t. Each node is identified by a unique and immutable identifier denoted by ID. A node $u \in V_t$ can send a message to a node $v \in V_t$ only if it knows the ID of node v. In a real-world network, these IDs could, e.g., be the nodes' IP addresses. This results in series of graphs $\mathcal{G} := (G_0, G_1, \dots)$ with $G_i = (V_i, E_i)$ and $E_t := \{(u, v) \mid u$ knows the ID of v in round $t\}$. We assume that a node can create edges to $O(\log n)$ different nodes in each round and can send $O(\text{polylog } n)$ bits via each edge.

Our research went in two directions that complement each other. First, we asked ourselves how to efficiently construct a robust overlay from any initial topology. Given any connected graph of n nodes representing our overlay, transform it into a low-diameter and high-expansion network. This protocol can be executed periodically to let the overlay recover from heavy but uncoordinated churn. Second, we assumed that the network already has a suitable topology but is attacked by a powerful adversary. This adversary tries to strategically disable nodes with crucial positions within the overlay. In this situation, the adversary's knowledge of the system and the system's reaction time, i.e., whether or not the nodes can detect if they get attacked, are crucial to the success probability of our defenses. We developed a nuanced model to study several types of adversaries and presented competitive protocols that are safe against powerful adversaries.

---

[16]Synchronicity is a standard assumption as nodes need to react to the adversary's changes promptly.

Result	Runtime	Init. Topology	Communication[a]
[AAC+05]	$O(d^{\mathrm{b}} + \log^2 n)$ w.h.p	Any	$O(\log n)$
[GHSS17][*]	$O(\log^2 n)$	Any	$O(d \log n)$
[GHS19][*]	$O(\log^{3/2} n)$ w.h.p	Any	$O(d \log n)$
[GHSW21][*]	$O(\log n)$ w.h.p	Any	$O(d \log n)$

   [a] Number of messages per node and round.
   [b] d denotes the initial graph's degree.
   [*] Supported by the CRC 901.

Table 1: *An overview of the overlay construction algorithms.*

Fast Construction of Overlays

To the best of our knowledge, the first overlay construction algorithm with polylogarithmic time and communication complexity that can handle (almost) arbitrary initial states has been proposed by Angluin et al. [AAC+05]. Here, the authors assume a weakly connected initial graph of degree d. If in each round, each node can send and receive at most d messages, and new edges can be established by sending node identifiers, their algorithm transforms the graph into a binary search tree of depth $O(\log n)$ in $O(d + \log^2 n)$ time, w.h.p. A low-depth tree can easily be transformed into many other topologies, and fundamental problems such as sorting or routing can be easily solved from such a structure. This idea has sparked a line of research investigating how quickly such overlays can be constructed. In the context of the CRC, we contributed three results. First, Gmyr et al. presented a deterministic $O(\log^2 n)$ time algorithm [GHSS17]. The algorithm's key idea is to maintain a series of so-called *supernodes*, which are groups of nodes that act in coordination. The algorithm operates in phases of $O(\log(n))$ rounds where each supernode merges with at least one of its neighboring supernodes in each phase. Thus, the size of each supernode, i.e., the number of nodes it contains, grows by a constant factor in each phase. This results in a deterministic runtime of $O(\log^2(n))$. Using randomization, we further improved this procedure to time $O(\log^{3/2} n)$, w.h.p. [GHS19]. Our main idea was to merge several clusters of supernodes to increase the growth in each phase. Finally, we further optimized the runtime to the optimal value of $O(\log(n))$ in [GHSW21]. This approach is different from *all* previous algorithms in that it does *not* use any form of clustering to contract large portions of the graph into supernodes. On a high level, our algorithm progresses through $O(\log n)$ iterations, where the next graph is obtained by establishing random edges on the current graph. These random edges are simply sampled by constant-length random walks, resulting in an extremely simple yet fast algorithm. Table 1 provides an overview of all contributions.

Efficient Maintenance of Overlays

Drees et. al developed our first approach to handling high churn in [DGS16]. The core idea was to use random walks to reorganize the network continuously. However, instead of just using random walks in a standard fashion, which would take $\Omega(\log n)$ communication rounds in graphs of polylogarithmic degree to sample nodes uniformly at random, they

combine random walks with *pointer jumping*. Pointer jumping, i.e., letting a node introduce its neighbors to its neighbors, is a well-known technique in the area of parallel computing, but to great surprise, it seems that it has never been combined with random walks so far. This rather simple trick exponentially improves the running time needed to sample nodes uniformly at random via random walks. We refer to the technique of combining random walks with pointer jumping to sample nodes from a network as *rapid node sampling*.

Based on rapid node sampling, Drees et al. developed algorithms that maintain the connectivity of a network under heavy churn and DoS attacks. Their algorithm organizes the nodes of a network into an expander and maintains connectivity under adversarial churn by an omniscient adversary with a constant churn rate. An important assumption underlying this result is that a node that is prescribed to leave the network by the adversary does not have to leave immediately but can remain in the network for another $O(\log \log n)$ rounds.

On the flip side, rapid node sampling cannot be used if one wants to grant the adversary access to even more recent information than $O(\log \log n)$ rounds. To overcome this restriction, Götte et al. proposed a trade-off in[GVS19]. This trade-off comes in the form of a (a, b)-late omniscient adversary that has almost up-to-date information about the network topology, but it is more outdated concerning all other aspects. In particular, it has full knowledge of the topology after a rounds and complete knowledge of messages, internal states, etc., after b rounds. In the real world, an adversary with similar properties could, e.g., be an agency eavesdropping on Internet exchange points. They can see *who* communicates based on the involved IP addresses but cannot decrypt the messages (or take longer to decrypt them).

The main contribution is a distributed overlay maintenance algorithm that completely rearranges the network every 2 rounds and can handle a $(2, O(\log n))$-late adversary. Furthermore, the algorithm allows routing a message to a logical address $p \in [0, 1)$ within $O(\log n)$ rounds. The algorithms are randomized and the results hold *w.h.p.* Instead of a regular expander, they use a structured overlay topology, namely, an extension of the Linearized DeBruijn Graph presented in Richa et al. [RSS11]. Götte et al. present a robust algorithm that minimizes the number of messages sent in every step. The approach uses several structural properties of the overlay as well as a careful analysis of non-independent events to ensure fast reconfiguration of the network.

Table 2 provides an overview of our results and compares them to previous and concurrent works. As one can see, our results compare favorably with regard of the churn rate and adversarial knowledge they tolerate.

3   Impact and Outlook

With our research on reputation systems and anonymous credentials we contributed significantly to the rapidly increasing research on cryptographic privacy-preserving techniques. Given the dramatic growth of online services, web shops, social media apps, and other data demanding tools these techniques will become ever more important in the future. Our research certainly had significant impact on basic scientific research. But the more important contribution of our research is probably in reducing the gap between theory and practice. Although many advanced cryptographic privacy-preserving techniques exist, they

Paper	Lateness[a]	Churn Rate[b]	Immediate
[AS18]	$(O(\log\log n), O(\log\log n))$	$(\alpha n, O(\log\log n))$	Yes
[AMM$^+$13]	$(O(\log n), O(\log n))$	$\left(O\left(\frac{n}{\log n}\right), O(\log n)\right)$	Yes
[DGS16]*	$(O(\log\log n), O(\log\log n))$	$(n - \frac{n}{\log n}, O(\log\log n))$	No[c]
[GVS19]*	$(2, O(\log n))$	$(\alpha n, O(\log n))$	Yes

[a] An adversary is (a, b)-late if it has full knowledge of the topology after a rounds and complete knowledge of all messages after b rounds.

[b] The churn rate is (C, T) if the adversary can perform C join/leaves in T rounds.

[c] Nodes remain in the network for additional $O(\log\log n)$ rounds.

* Supported by the CRC 901.

Table 2: *Overview of different models in the literature*

are rarely implemented in prototypes or even used in commercial applications. This is mainly due to two factors: advanced cryptographic techniques are often believed to be too inefficient and cumbersome, and cryptographic techniques are difficult to implement from scratch. To overcome these misconceptions and impediments we complemented our basic research by two more applied approaches:

1. Based on our research on updatable credentials we designed a so-called incentive systems and implemented it from scratch.

2. We built a cryptographic open-source Java library, called cryptimeleon, providing basic cryptographic primitives that are the backbone of many privacy-preserving techniques.

Since the incentive system is discussed in detail in the section on transfer project T2, we concentrate on cryptimeleon. The library allows users to build complex privacy-preserving primitives in the so-called bilinear group setting, currently the most powerful and efficient setting for cryptography (although not post-quantum secure, see below). It provides a general framework for the construction of primitives and a number of important basic primitives such as hash functions, pseudorandom functions and Schnorr-type zero-knowledge proofs. A detailed description of the library can be found in [BEHF21]. Although cryptimeleon is mainly targeted towards researchers in cryptography (which already use it in increasing numbers) we believe that it can also form the foundation for a library targeted at more general users.

Our research on reputation systems raises many questions for future research, e.g., how to realize such systems in a decentralized form. But from our perspective, the most pressing problem is to bring many more privacy-preserving techniques to the post-quantum world. Ever since the seminal algorithm of Peter Shor, we know that currently used cryptographic techniques are susceptible to quantum attacks. The development of quantum computers has seen tremendous progress in the last years. Although it is still unclear if and when quantum computers will be built on which Shor's algorithm can be implemented, we have to prepare our security infrastructure for this event: hence the standardization efforts for post-quantum secure cryptography by the NIST and other organizations worldwide.

For basic cryptographic primitives such as encryption schemes and digital signatures, we know many (hopefully) post-quantum secure constructions. For more advanced privacy-preserving techniques, such as reputation systems and anonymous credentials, the situation is quite different. Additionally, post-quantum cryptography tends to be technically much more challenging than classical cryptography. Therefore, it is even more urgent for cryptographers to provide users with easy to use post-quantum secure cryptographic libraries. In the context of verifiable random functions, our work has answered fundamental open questions, but also raised some further questions that are still open. While our constructions of VRFs are already much more efficient than previous constructions, they are still much less efficient than constructions in the ROM. Hence, an important open question is whether there are VRFs that are secure (under standard assumptions) in the standard model and that are as efficient as VRFs whose security proof requires the ROM. Since our techniques used to construct VRFs were also applicable to IBE, one can similarly ask the question whether standard model IBEs can be as efficient as IBEs that are proven secure in the ROM. One can also ask whether there exist IBEs that are secure in the standard model and as efficient as schemes that are proven secure in the ROM. With respect to tightness of security reductions for VRFs, our construction is proven secure under a so-called q-type assumption. These types of assumption have the negative aspect that they get stronger the larger q becomes [Che10]. For these reasons q-type assumptions are not considered standard assumptions. Thus, it would be preferable to achieve tightness with a security proof that relies on a standard assumption. Whether there are VRFs that can be proven secure under a standard assumption and achieve optimal tightness is another fundamental research question in this domain.

Our results on overlays and data storage are (asymptotically) optimal or at least very close to their optimal solution, i.e., within polylogarithmic factors, concerning time and message complexity. Further, our random walk-based algorithms are heavily inspired by those used in practice in big P2P networks such as the one of Bitcoin, giving them a sound theoretical foundation. However, throughout all our contributions, we only considered benign failures of nodes. The nodes may unexpectedly crash but generally behave correctly and follow the protocol. For example, they do not intentionally alter any data item they store or introduce invalid identifiers to the system. So, a natural follow-up question is how our algorithms could also be extended to tolerate nodes exhibiting this and other malicious behavior. More precisely, we want to consider so-called byzantine nodes that deviate from the protocol and behave arbitrarily. Such nodes are not a niche phenomenon but are a very common threat in internet-scale applications: First, not everyone connected to the internet is trustworthy and may be controlled by an adversarial party with their own malicious goals. Second, even honest and well-protected nodes may be hacked by an attacker if the stakes are high enough.

A typical way to deal with byzantine nodes is to use so-called quorums. These are (randomly selected) subsets of nodes that act in coordination. In particular, the honest nodes outnumber the byzantine nodes in each quorum, so potentially malicious actions can be *overruled*. However, these quorums introduce a massive overhead because data must be replicated and passed to all members. Further, for all messages between quorums, the members of each quorum need to agree on the content. Reaching an agreement in the presence of byzantine nodes is notorious for being time and/or bandwidth-consuming as it typically requires all-to-all communication. To add insult to injury, using quorums is

a proactive approach. This means that we pay the high cost of maintaining the quorums even if there is no byzantine behavior (and we could have used our original algorithms). With our current models and assumptions, there seems to be no way around using quorums if one looks for proactive approaches. Because of this, we want to shift our attention to *reactive* approaches and design algorithms that are at least resource competitive. This means that the cost of executing the algorithm, e.g., the latency and message complexity, is directly related to the severity of the byzantine attack. The caveat of this approach is that the system needs some mechanism to recover from an attack, e.g., to retrieve dropped data items or reconnect parts of the overlay. This mechanism could be implemented through a trusted third party, e.g., a cloud provider, that offers these services for a price. Whenever the honest nodes notice byzantine behavior, they query the third party to repair the system. For example, this third party could store data items. If an honest node detects that data items have been dropped or tampered with in the P2P network, it retrieves a *fresh* copy from the trusted party. Ideally, the number of queries is linear in the number of tampered data items. However, there are many open questions about this approach that require nuanced answers: for instance, how the cost of queries is measured and what the exact capabilities of the trusted party are. In particular, the trusted party must be implementable in practice and not be too powerful to keep the problem interesting. We plan to investigate all these questions in the future.

Bibliography

[AAC⁺05]   Angluin, D.; Aspnes, J.; Chen, J.; Wu, Y.; Yin, Y.: Fast Construction of Overlay Networks. In: *Proc. of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 2005, pp. 145–154

[AMM⁺13]   Augustine, J.; Molla, A. R.; Morsy, E.; Pandurangan, G.; Robinson, P.; Upfal, E.: Storage and search in dynamic peer-to-peer networks. In: *Proc. of SPAA*. 2013, pp. 53–62

[AS18]   Augustine, J.; Sivasubramaniam, S.: Spartan: A Framework For Sparse Robust Addressable Networks. In: *Proc. of IPDPS*. 2018, pp. 1060–1069

[BB18]   Blömer, J.; Bobolz, J.: Delegatable Attribute-Based Anonymous Credentials from Dynamically Malleable Signatures. In: *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*. Vol. 10892. Lecture Notes in Computer Science. Springer, 2018, pp. 221–239

[BBB⁺18]   Bemmann, K.; Blömer, J.; Bobolz, J.; Bröcher, H.; Diemert, D.; Eidens, F.; Eilers, L.; Haltermann, J.; Juhnke, J.; Otour, B.; Porzenheim, L.; Pukrop, S.; Schilling, E.; Schlichtig, M.; Stienemeier, M.: Fully-Featured Anonymous Credentials with Reputation System. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES*. ACM, 2018, 42:1–42:10

[BBDE19]   Blömer, J.; Bobolz, J.; Diemert, D.; Eidens, F.: Updatable Anonymous Credentials and Applications to Incentive Systems. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. ACM Press, 2019, pp. 1671–1685

[BBS04]   Boneh, D.; Boyen, X.; Shacham, H.: Short Group Signatures. In: *Advances in Cryptology – CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55

[BCC⁺09]   Belenkiy, M.; Camenisch, J.; Chase, M.; Kohlweiss, M.; Lysyanskaya, A.; Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: *Advances in Cryptology – CRYPTO 2009*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 108–125

[BEHF21]   Bobolz, J.; Eidens, F.; Heitjohann, R.; Fell, J.: *Cryptimeleon: A Library for Fast Proto-typing of Privacy-Preserving Cryptographic Schemes*. Cryptology ePrint Archive, Report 2021/961. https://eprint.iacr.org/2021/961. 2021

[BEJ18]   Blömer, J.; Eidens, F.; Juhnke, J.: Practical, anonymous, and publicly linkable universally-composable reputation systems. In: *Cryptographers' Track at the RSA Conference*. Springer. 2018, pp. 470–490

[BJK15]   Blömer, J.; Juhnke, J.; Kolb, C.: Anonymous and Publicly Linkable Reputation Systems. In: *FC 2015: 19th International Conference on Financial Cryptography and Data Security*. Vol. 8975. Lecture Notes in Computer Science. Springer, 2015, pp. 478–488

[BPS+17]   Busom, N.; Petrlic, R.; Sebé, F.; Sorge, C.; Valls, M.: A privacy-preserving reputation system with user rewards. In: *Journal of Network and Computer Applications* 80 (2017)

[BR09]   Bellare, M.; Ristenpart, T.: Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme. In: *Advances in Cryptology - EUROCRYPT 2009*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 407–424.

[Can01]   Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: *42nd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 2001, pp. 136–145

[CDD17]   Camenisch, J.; Drijvers, M.; Dubovitskaya, M.: Practical UC-Secure Delegatable Credentials with Attributes and Their Application to Blockchain. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. ACM Press, 2017, pp. 683–699

[CG08]   Camenisch, J.; Gross, T.: Efficient attributes for anonymous credentials. In: *ACM CCS 2008: 15th Conference on Computer and Communications Security*. ACM Press, 2008, pp. 345–356

[CGH04]   Canetti, R.; Goldreich, O.; Halevi, S.: The random oracle methodology, revisited. In: *J. ACM* 51 (2004), no. 4, pp. 557–594.

[Che10]   Cheon, J. H.: Discrete Logarithm Problems with Auxiliary Inputs. In: *J. Cryptol.* 23 (2010), no. 3, pp. 457–476.

[CKS10]   Camenisch, J.; Kohlweiss, M.; Soriente, C.: Solving Revocation with Efficient Update of Anonymous Credentials. In: *SCN 10: 7th International Conference on Security in Communication Networks*. Vol. 6280. Lecture Notes in Computer Science. Springer, 2010, pp. 454–471

[CL01]   Camenisch, J.; Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: *Advances in Cryptology – EUROCRYPT 2001*. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 93–118

[CL02]   Camenisch, J.; Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: *Advances in Cryptology – CRYPTO 2002*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 61–76

[CL04]   Camenisch, J.; Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: *Advances in Cryptology – CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 56–72

[CL19]   Crites, E. C.; Lysyanskaya, A.: Delegatable Anonymous Credentials from Mercurial Signatures. In: *Topics in Cryptology – CT-RSA 2019*. Vol. 11405. Lecture Notes in Computer Science. Springer, 2019, pp. 535–555

[CLNR14]   Camenisch, J.; Lehmann, A.; Neven, G.; Rial, A.: Privacy-Preserving Auditing for Attribute-Based Credentials. In: *ESORICS 2014: 19th European Symposium on Research in Computer Security, Part II*. Lecture Notes in Computer Science. Springer, 2014, pp. 109–127

[DGS16]   Drees, M.; Gmyr, R.; Scheideler, C.: Churn- and DoS-resistant Overlay Networks Based on Network Reconfiguration. In: *Proc. of SPAA*. 2016, pp. 417–427

[Eik16]   EIKEL, M.: Insider-resistant distributed storage systems. PhD thesis. University of Pader-
            born, 2016.

[ES15]   EIKEL, M.; SCHEIDELER, C.: IRIS: A Robust Information System Against Insider DoS
            Attacks. In: *ACM Trans. Parallel Comput.* 2 (2015), no. 3, 18:1–18:33

[ESS14]   EIKEL, M.; SCHEIDELER, C.; SETZER, A.: RoBuSt: A Crash-Failure-Resistant Distributed
            Storage System. In: *Principles of Distributed Systems - 18th International Conference,
            (OPODIS)*. 2014, pp. 107–122

[GHS19]   GÖTTE, T.; HINNENTHAL, K.; SCHEIDELER, C.: Faster Construction of Overlay Networks.
            In: *International Colloquium on Structural Information and Communication Complexity
            (SIROCCO)*. Springer. 2019, pp. 262–276

[GHSS17]   GMYR, R.; HINNENTHAL, K.; SCHEIDELER, C.; SOHLER, C.: Distributed Monitoring of Network
            Properties: The Power of Hybrid Networks. In: *Proc. of the 44th International Colloquium
            on Automata, Languages, and Programming (ICALP)*. 2017, 137:1–137:15

[GHSW21]   GÖTTE, T.; HINNENTHAL, K.; SCHEIDELER, C.; WERTHMANN, J.: Time-Optimal Construction
            of Overlay Networks. In: *PODC '21: ACM Symposium on Principles of Distributed
            Computing*. ACM, 2021, pp. 457–468.

[GVS19]   GÖTTE, T.; VIJAYALAKSHMI, V. R.; SCHEIDELER, C.: Always be Two Steps Ahead of Your
            Enemy. In: *2019 IEEE International Parallel and Distributed Processing Symposium,
            IPDPS*. IEEE, 2019, pp. 1073–1082.

[JKN21]   JAGER, T.; KUREK, R.; NIEHUES, D.: Efficient Adaptively-Secure IB-KEMs and VRFs
            via Near-Collision Resistance. In: *Public-Key Cryptography - PKC 2021 - 24th IACR*.
            Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 596–626.

[JN19]   JAGER, T.; NIEHUES, D.: On the Real-World Instantiability of Admissible Hash Functions
            and Efficient Verifiable Random Functions. In: *Selected Areas in Cryptography - SAC
            2019*. Vol. 11959. Lecture Notes in Computer Science. Springer, 2019, pp. 303–332.

[MBB+15]   MELARA, M. S.; BLANKSTEIN, A.; BONNEAU, J.; FELTEN, E. W.; FREEDMAN, M. J.: CONIKS:
            Bringing Key Transparency to End Users. In: *24th USENIX Security Symposium, USENIX
            Security 15*. USENIX Association, 2015, pp. 383–398.

[MSBM22]   MIR, O.; SLAMANIG, D.; BAUER, B.; MAYRHOFER, R.: *Practical Delegatable Anonymous Cre-
            dentials From Equivalence Class Signatures*. Cryptology ePrint Archive, Report 2022/680.
            https://eprint.iacr.org/2022/680. 2022

[NF06]   NAKANISHI, T.; FUNABIKI, N.: A Short Verifier-Local Revocation Group Signature Scheme
            with Backward Unlinkability. In: *Advances in Information and Computer Security*. Vol. 4266.
            LNCS. Springer, 2006, pp. 17–32

[Nie21a]   NIEHUES, D.: More Efficient Techniques for Adaptively-Secure Cryptography. PhD thesis.
            University of Wuppertal, Germany, 2021.

[Nie21b]   NIEHUES, D.: Verifiable Random Functions with Optimal Tightness. In: *Public-Key Cryp-
            tography - PKC 2021 - 24th IACR*. Vol. 12711. Lecture Notes in Computer Science.
            Springer, 2021, pp. 61–91.

[PS16]   POINTCHEVAL, D.; SANDERS, O.: Short Randomizable Signatures. In: *Topics in Cryptology –
            CT-RSA 2016*. Vol. 9610. Lecture Notes in Computer Science. Springer, 2016, pp. 111–126

[RSS11]   RICHA, A. W.; SCHEIDELER, C.; STEVENS, P.: Self-Stabilizing De Bruijn Networks. In: *Proc.
            of SSS*. 2011, pp. 416–430

[ZWC+16]   ZHAI, E.; WOLINSKY, D. I.; CHEN, R.; SYTA, E.; TENG, C.; FORD, B.: AnonRep: Towards
            Tracking-Resistant Anonymous Reputation. In: *NSDI*. 2016, pp. 583–596

Subproject C2:
On-The-Fly Compute Centers I: Heterogeneous Execution Environments

Tim Hansmeier[1], Tobias Kenter[2], Marius Meyer[2], Heinrich Riebler[2],
Marco Platzner[1], Christian Plessl[2]

1 Department of Computer Science, Paderborn University,
Paderborn, Germany
2 Paderborn Center for Parallel Computing and
Department of Computer Science, Paderborn University,
Paderborn, Germany

1 Introduction

Subproject C2 investigated the execution of configured IT services in OTF Compute Centers with heterogeneous computing nodes combining CPUs, GPUs, and FPGAs. The key idea in the subproject is that a service can exist in multiple variants that are specifically tailored for different processor or accelerator architectures. While the execution of these variants leads to the same functional behavior, the non-functional properties, such as energy consumption or latency, may differ considerably. We can exploit this fact by creating variants of services for different hardware architectures at compile time and chosing the optimal variant at runtime according to resource availability to improve performance or efficiency, for example.

In Subproject C2, we have developed methods for this purpose. Specifically, we have studied, how we can create programming models that enable and exploit dynamic dispatching of services (which are themselves part of composed services) to different execution resources; how we can model, optimize, and empirically validate the benefits of dynamic dispatching of services; and how we can develop novel hardware architectures and runtime systems for increasing the effectiveness of this approach.

Over the three funding periods for CRC 901, we have studied the aforementioned idea of dynamic dispatching of services to heterogeneous resources, putting a different emphasis in each of the funding periods, which is summarized in the following.

In the *first funding period*, fundamental architectures and basic mechanisms for heterogeneous migration of services between different computing resources were developed. Heterogeneous migration includes transmodal migration between software and hardware as a special case. We have implemented a system based on POSIX threads with a scheduler for heterogeneous systems and a programming model tuned to it. The approach supports

tim.hansmeier@uni-paderborn.de (Tim Hansmeier), kenter@uni-paderborn.de (Tobias Kenter), marius.meyer@uni-paderborn.de (Marius Meyer), heinrich.riebler@uni-paderborn.de (Heinrich Riebler), platzner@ubp.de (Marco Platzner), christian.plessl@uni-paderborn.de (Christian Plessl)

the implementation and execution of OTF services on different target architectures by a checkpointing mechanism. This makes it possible to interrupt a running service, save its state, and migrate it.

Furthermore, basic concepts for an on-the-fly hardware acceleration approach have been developed. Here, implementations for the different architectures do not have to be created manually but are generated automatically for a limited set of application classes. These methods open up fundamentally new possibilities for dynamically allocating services to the available computing resources and migrating between resources as needed such that application- and system-level goals can be optimized, e.g., throughput, energy consumption, etc. The necessary adaptation of the application to the given programming model and the requirement for architecture-independent checkpoints, however limit the usability and productivity of the approach.

For the *second funding period*, we have therefore chosen programmability and efficiency as the focus topics. By aligning our programming model with OpenCL as a standardized programming interface for heterogeneous computing, the effort required to program heterogeneously migratable services was significantly reduced. Building on this programming model, scheduling algorithms were developed to optimize runtime and energy consumption. Programming for heterogeneous migration is elaborated in Section 2, and a new runtime system able to automatically generate OpenCL accelerator code from sequential CPU code is discussed in Section 5. For scheduling and migration decisions, it is useful to have as precise information as possible about how well individual services are suited for execution on different target architectures. For the necessary off-line characterization of services, we have developed the Ampehre framework, which allows precise measurements of many system parameters of the heterogeneous computing node. The Ampehre measurement framework has been employed to create a highly accurate energy model for task execution on heterogeneous compute nodes [LP18] and was instrumental in developing schedulers utilizing heterogeneous task migration to minimize runtime and energy [LP17], [LP20]. A more detailed description of Ampehre is given in Section 4.

To improve the efficiency of on-the-fly acceleration with FPGAs, overlay architectures were developed that trade off between maximum specialization and full programmability but can be configured much faster. This allows FPGAs to largely avoid the long synthesis and implementation times associated with full specialization. To investigate this approach, hand-designed overlays have been studied and methods for automatically configuring overlays have been developed. This approach is presented in Section 3.

Finally, the focus of the *third funding period* was on mastering the complexity of modern architectures and runtime systems in heterogeneous OTF compute center architectures, where compute nodes must run composed services with varying requirements and optimization goals. The development of increasingly complex runtime systems with centralized scheduling no longer seems promising for such systems. The heterogeneity in the OTF compute centers leads to large amounts of diverse information and optimization goals, which makes designing a centralized scheduler an infeasibly complex task. Instead, we studied concepts of self-aware computing to provide runtime systems with an increased degree of autonomy and learning capability. We experimented with learning classifier systems, in particular XCS, as algorithmic methods for achieving the required learning capability for heterogeneous compute nodes [Han21]. We extended XCS to allow them to adapt their parameters at runtime [HKP20a], which is highly effective in dynamic environ-

ments [HKP20b]. Furthermore, we empirically evaluated different strategies for switching between exploration and exploitation [HP21] and presented an approach for providing safety guarantees [HP22] for XCS. We also came up with an embedded implementation of XCS [HBP22] to evaluate its resource consumption.

We extended our consideration from a single heterogeneous compute node, each with one FPGA, to an entire cluster of such compute nodes. To exploit this kind of infrastructure, models and procedures to partition the composed services onto a multi-FPGA cluster in a meaningful way are required. Therefore, we developed an OpenCL benchmark suite [MKP20] to determine and capture the relevant performance characteristics for the execution of services on such a system, which can serve as a basis for more accurate models for dynamic composition of services in the spirit of the OTF concept. Major aspects were the configurability of the benchmarks [MKP22] and the utilization of direct and highly efficient FPGA-to-FPGA interconnect options in addition to a classical architectural approach in which FPGAs communicate only via CPUs [MKP23]. The benchmarks in the suite are designed to support these various communication approaches and produce comparable performance results for all considered communication infrastructures.

2 Programming for Heterogeneous Migration

Computing nodes are increasingly heterogeneous and augment CPUs with accelerator technologies such as GPUs and FPGAs. To benefit from such a computing environment, developers must identify hotspots or tasks in their applications, port those to the available accelerators, and finally optimize them to achieve high performance. Additionally, scheduling techniques are needed that distribute the workload of one or several applications to the heterogeneous resources, subject to an optimization objective such as minimization of runtime or energy consumption.

The introduction of OpenCL as a programming language greatly simplified the use of accelerator technologies. With OpenCL compilers available for CPUs, GPUs, and even FPGAs, application code is basically executable on all these resources without any additional porting effort. However, since OpenCL is not performance-portable, developers must still optimize their task implementations to achieve good performance for the different resources.

Current accelerators rely on a run-to-completion execution model, where a task assigned to an accelerator computes there until termination. This is in strong contrast to CPU-based computing, where operating systems provide preemptive multitasking, and might severely impact system performance since a running task cannot be migrated to a better-suited resource in a later execution phase.

In this section, we give an overview of our novel OpenCL-based programming framework that overcomes the limitations of the run-to-completion approach. We introduce a programming pattern and execution model for tasks that allow us to migrate them between the resources of a heterogeneous compute node at predefined states without losing their computational progress. While we focus on OpenCL, the approach is more general and supports programming languages with host-centric execution models, also including OpenMP, OpenACC, CUDA, and the Maxeler MaxJ hardware description language. Developers only need to provide functionally identical task implementations for the resources. Furthermore,

our work includes an interface for inter-process communication between the tasks and a scheduler framework. Using this interface, schedulers can decide and execute task-resource assignment, including heterogeneous migration.

Applications following a host-centric programming style are bipartite: The *host code* is responsible for resource management, which includes allocating local memory on the accelerator, transferring data to and from the accelerator, and triggering computations. The computations on the accelerator are denoted as *kernel code*. The CPU plays a special role since it is the host and can at the same time also execute kernel code.

In our approach, we store relevant task state information in memory and transfer this information between the host memory and an accelerator's local memory. Through such state transfers, migration can be implemented even between very diverse resources such as FPGAs and GPUs. This technique is often referred to as *checkpointing* and poses two challenges: First, task developers or automated tools need to identify checkpoints in application tasks that can be mapped to other resources in order to continue the task computation without loss in their computational progress. Moreover, minimal task states are favorable since state transfers are expensive and constitute overhead. Second, the checkpointing frequency must be carefully selected to balance between the overhead incurred by checkpointing and the ability of being able to quickly migrate when needed.

A possible method to enable task migration by checkpointing is adapting the *loop strip mining* transformation to a task's kernel code. The loop of a data-parallel kernel is split into an outer and an inner loop, which is vectorizable. The outer loop is then run as host code, and the adjusted kernel comprising the inner loop is called from the host. This way, the adjusted kernel works on blocks of data successively and after each kernel execution, i.e., an iteration of the outer loop, the checkpoint can be transferred. Since the inner loop is kept in vectorized form, a checkpointed task implementation can provide high performance for data-parallel tasks if the *checkpoint distance* is sufficiently large.

Our programming pattern for heterogeneous task migration supports checkpointing and comprises five stages:

1. The *bookkeeping* stage is the task preparation stage, where we allocate memory space in the host memory and read input data from the hard disk or the network interface. This stage is resource-independent and therefore involves only activities handled by the CPU.

2. The *init* stage allocates memory space in the local memory of the accelerator and transfers the checkpoint to this memory.

3. The *compute* stage executes the kernel code. Each time the compute stage is called, the kernel processes the next block of data and stores its progress as updated checkpoint. Furthermore, the kernel must be able to report its computational progress to enable the host to keep track of the overall task computation.

4. The *fini* stage is the counterpart of the init stage and transfers the checkpoint back into host memory before releasing the accelerator device.

5. The *cleaning* stage is the final one and the counterpart of the bookkeeping stage, as it writes the computation results to the hard disk or the network interface. This stage is resource-independent and thus exclusively executed on the host processor.

Based on this pattern, a task is migrated from resource A to resource B by calling the fini stage for the task implementation running on resource A followed by executing the init stage of the task implementation for resource B. The resulting *migration overhead* comprises two parts: The first and frequently dominating part is the time to transfer the checkpoints during the fini and init stages. The second part includes additional steps for preparing the target resource, such as the reconfiguration of an FPGA device.

We explain the *lifecycle* of a migratable task using the example shown in Figure 39, which lists the pseudo code for the four major software components involved. `main.cpp` instantiates `ExampleOCL` configured for CPU usage and resource-specific implementations for GPU and FPGA. `ExampleOCL` itself loads the checkpointed OpenCL kernel `example` and compiles it for execution on the CPU. The kernel illustrates checkpointing by iterating over successive sections of a strip-mined loop, with `iters_per_checkpoint` specifying the size of data processes per kernel execution. The task's progress can be determined by comparing the `progress` counter with the `num_of_checkpoints`. Note the three resource-specific stages implemented in `ExampleOCL`. The init and fini stages are transferring the checkpoint between the host memory and the local memory of the device. Since host memory and CPU-related memory are identical, the checkpoint is not copied. The `compute` method is then working on the checkpoint by only reading and writing data in the local memory. After adding all resource-specific implementations of the task in `main.cpp`, the task executor `TaskExec` is called. The pseudo code in Figure 39 also illustrates the execution of `execute_online()`, interacting with a scheduler connected via Inter-Process Communication (IPC).

The code listed in Figure 39 correlates to the task lifecycle depicted as a flowchart in Figure 40. The dotted shapes clarify the mapping between the pseudo code and the flowchart. The first activity is calling the `bookkeeping` method, which is executed by the host processor and prepares the task for execution. Then, the task enters an execution loop where it remains until the entire data is processed, i.e., the task execution state is `FINISHED`. The first step in the execution loop is to wait for resource assignment, which is implemented as a blocking IPC receive call that returns the assigned resource from the scheduler. Next, the `init` method of the chosen task implementation is called and the current checkpoint is copied into the target local memory. The following do-while loop iteratively calls `compute` and informs the scheduler about the task execution progress. While `compute` actually executes resource-specific kernels on the CPU or accelerators, denoted by dark blue, red, and green colored box fillings, the `init` and `fini` box fillings are kept in light colors to depict that the devices are active by copying checkpoint data or reconfiguring the FPGA.

After the task execution progress has been sent to the scheduler, the task execution state is checked. In case the task execution has finished, we exit the do-while loop and call `fini` for the current task implementation, release the resource, and finally execute `cleaning`. If the task execution has not been finished yet, we communicate with the scheduler to figure out whether a task must be migrated. If so, the fini stage is called, the resource is released, and a new resource assignment is requested in the next iteration of the execution loop. After a new resource has been assigned, the task again calls the `init` method for the new resource.

The programming framework has been implemented in C++ and allows an easy integration of new computing resources by overriding corresponding class methods. Based on the

Figure 39: *Major software components for a migratable task.*

programming framework, two schedulers for heterogeneous compute nodes have been realized that demonstrate the potential of heterogeneous migration in terms of runtime and energy minimization. In [LP17], the reMinMin scheduler has been presented based on a static list scheduling approach for energy minimization. In [LP20], MigHEFT focused on scheduling migratable task graphs to heterogeneous resources.

3   Analyzing FPGA Overlays as Target for OTF Hardware Accelerator Generation

Overlays are configurations for FPGAs that are not fixed to a specific task, but instead provide a limited form of programmability, more abstract than that of the underlying FPGA fabric. Compared to highly optimized application-specific libraries, overlays enable significantly more applications as candidates for FPGA acceleration in an OTF context, because overlays can be more broadly applied. They thus provide a purchasing argument for FPGAs for OTF compute centers that need to aim for good utilization of their hardware over time. Also, in comparison to the synthesis of FPGA designs from high-level language code, where OpenCL and recently SYCL are particularly promising as a description language, overlays can help to avoid the extremely long synthesis runtimes of several hours
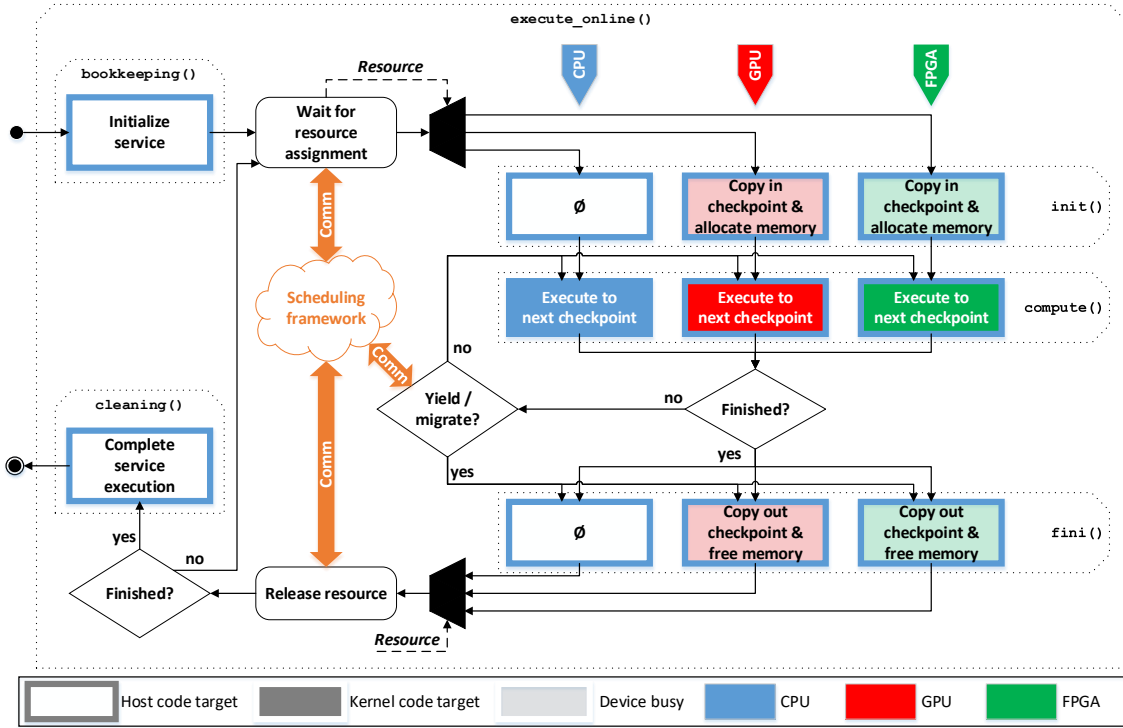
Figure 40: *Lifecycle of a migratable task.*

up to days, which are typical for FPGAs. Paired with suitable compilation approaches, they can also reduce the demand for manual development or optimization ahead of OTF deployment.
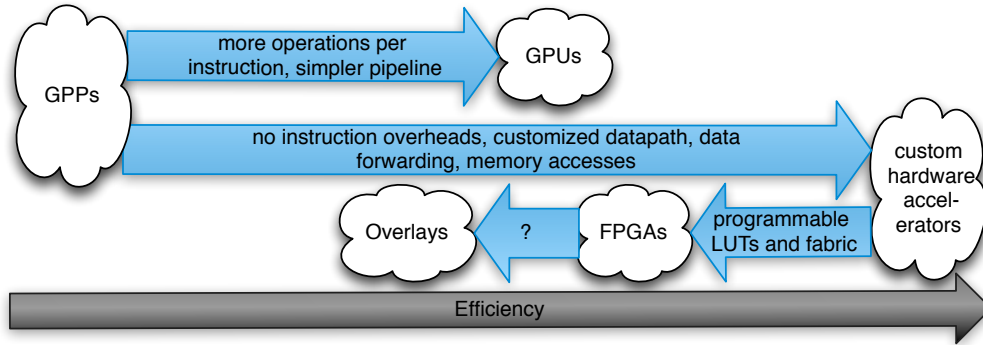
Figure 41: *Qualitative illustration on the impact of architecture features of accelerators on efficiency. Quantifying the performance overheads of FPGA overlays was the central research question of this contribution.*

We distinguish between processor-like instruction-programmable overlays and structurally programmable or configurable overlays. For both approaches, diverse architectures have been presented that gain their efficiency from various combinations of parallelism, pipelining, and targeted data access and reuse. These are already being investigated in the academic environment from various aspects such as productivity, portability, and scalability. For instruction-programmable overlays, it was however largely unclear until our work how close the performance of such architectures comes to that of fully specialized FPGA

implementations. Figure 41 illustrates the context of this research question in comparison to alternative accelerator architectures.

To answer this question, we have implemented a diverse set of computational tasks with identical interfaces on an overlay-based FPGA system and with highly specialized FPGA designs. The tasks here are all runtime-intensive steps (often referred to as kernels) from an application to compute stereo correspondence, which in turn is the most important and costly intermediate step to compute depth from a pair of stereo images. By following the best quality published algorithm [MSZ⁺11] in this area until recently, we achieve, on one hand, a comparison between overlay and specialized kernels that is not biased by FPGA-specific optimizations on the algorithm level and, on the other hand, the most accurate stereo matching implementation with FPGA acceleration to this time. In contrast, other FPGA implementations in this area (e.g. [SHW⁺14; JM14; TLLA14]) adapt the work steps and their sequence to the target architecture to varying degrees, thus achieving higher performance or lower space requirements on the FPGA with reduced result quality.

Due to the availability of suitable development tools and runtime environments to effectively implement the respective approaches with overlay and specialized kernels, the two approaches were implemented on two different target platforms: The Convey HC-1 with an instruction-programmable FPGA overlay with vector architecture on one hand and the Maxeler MPC-X platform with its own description language for highly specialized dataflow kernels on the other. Both represent state-of-the-art systems with a high-performance server processor and FPGA accelerator at the respective time of acquisition.
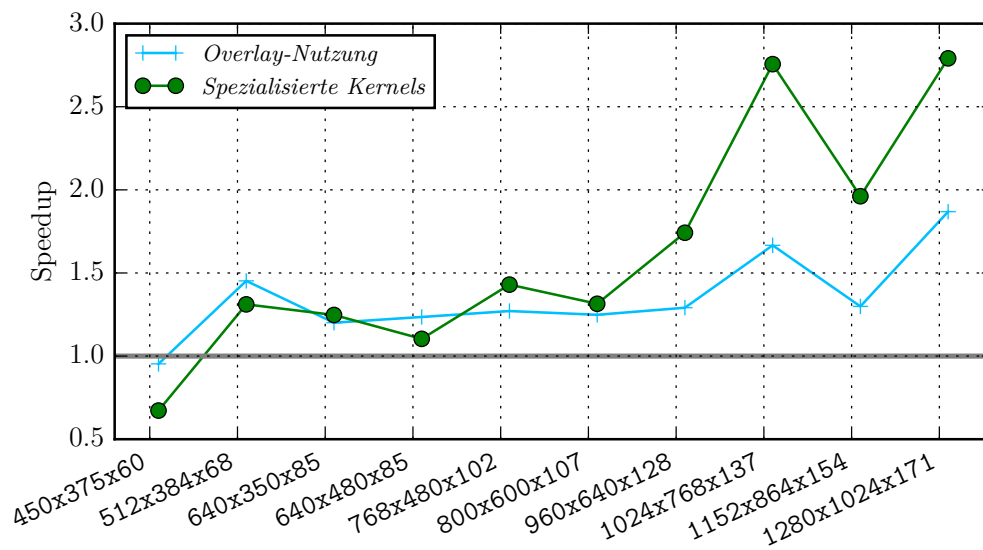
Figure 42: *Speedups of fully integrated stereo matching implementations on two systems with FPGA accelerators. These measurements include overheads for reconfiguration and data transfers, which favors the overlay architecture for small problem sizes in comparison to the specialized kernels.*

On both target platforms, the fully integrated computation of the stereo correspondence is executed by offloading the runtime-intensive kernels to the respective FPGA accelerator. Preparatory and management steps remain on the respective main processor. The runtimes for data transfers, synchronization and, in the case of the specialized kernel designs, recon-

figuration included in this execution model limit the achievable performance. Nevertheless, illustrated in Figure 42, both accelerator platforms achieve performance advantages over the powerful main processor of the Maxeler MPC-X platform for most input sizes of the application.

To quantify the conceptually driven performance differences between using an instruction-programmable overlay and fully specialized FPGA designs, we had to factor out influences of the specific platforms and their runtime environments. In [Ken16] and [KSP15], these steps are explained in detail. Subsequently, it can be shown that using the overlay yields on average about a factor of 3 less isolated kernel performance than FPGA implementations fully adapted to the task. In return, for the overlay, the runtimes of the tools used for translation or synthesis are several orders of magnitude shorter, amounting to only seconds instead of hours or even days. Overall, the high productivity required to profit from FPGA acceleration in the OTF context is not achieved by tool runtimes alone, but also depends on programming patterns (see also Section 2) or automated tools (see also Section 5) for code generation or overlay configuration generation and offloading to accelerators.

Thus, in the end, the decision between fully specialized FPGA kernels and overlay usage is similar to the decision between ASICs and FPGAs: Given sufficient development time, budget, expertise, and given a sufficiently high application demand, it will typically pay off to fully specialize. However, if any of these preconditions is not met—as can often be the case in OTF scenarios—overlays can provide an interesting alternative, at a performance cost that we now understand better.

4   AMPEHRE: An Extensible Measurement Framework for Heterogeneous Compute Nodes

Application performance profiling is a major step in software development. Based on hardware performance counters provided by the target devices and on timing information, developers gain knowledge about runtime behavior in terms of metrics such as the number of executed instructions, cache misses, page-faults, or statistics about called functions. Understanding runtime behavior is instrumental for optimizing performance. Examples for widely-used performance analysis tools are the open-source tools `Perf`, `IgProf`, and `Likwid`, and the vendor-specific tools Intel VTune Amplifier or the Nvidia GPU development IDE Nvidia Nsight and command-line tool `nvprof`. With the introduction of the Running Average Power Limit (RAPL) interface for Intel CPUs, developers are also able to perform energy measurements on CPUs.

A shortcoming of most existing tools is their lack of an easy-to-use and extensible application programmer interface (API) that allows user applications to read performance and energy data comparable across different resource types. The Performance Application Programming Interface (PAPI) project has been developed to help solve this issue. Particularly with the PAPI version 5 release, developers are able to add capabilities for power or temperature analysis by implementing so-called *PAPI components*, extending PAPI to new platforms and other sensor types. PAPI provides a unified API that hides the underlying device-specific measuring procedures when reading power, energy, and temperature sensors. But, even with PAPI, the retrieved data must be interpreted to gain semantically comparable measurements results across resource boundaries.
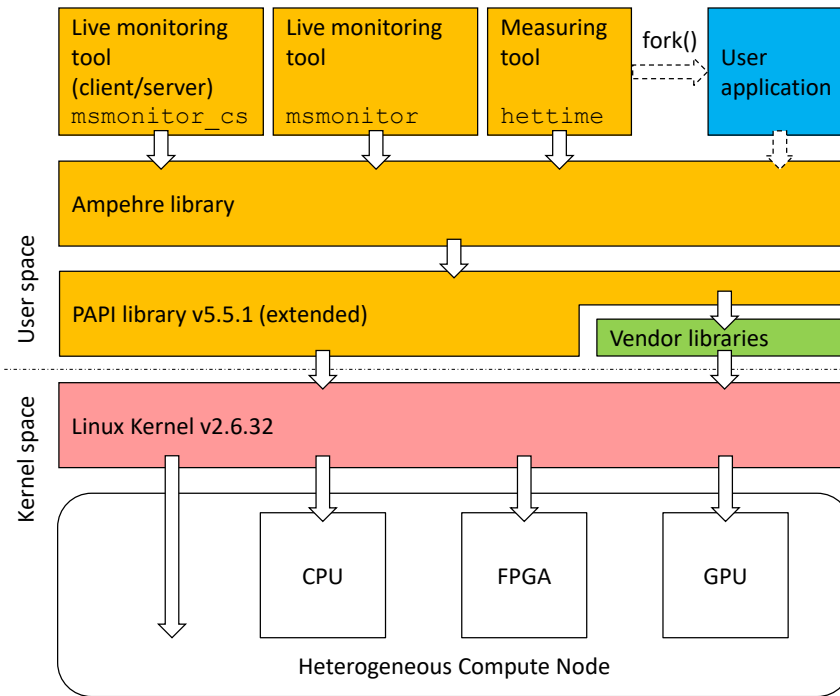
Figure 43: *Ampehre architecture (taken from [LWP18]). Blocks in orange denote components we have implemented or extended.*

To improve on this situation, we have developed the measurement framework *Ampehre*, short for *Accurately Measuring Power and Energy for Heterogeneous Resource Environments* [LKEW]. Ampehre is designed for heterogeneous high-performance compute nodes running Linux and (i) allows an easy integration into applications by providing a clear API covering all resource types, (ii) is extensible to new resources and sensors through the use of PAPI, and (iii) is available as open source.

Figure 43 presents the architecture of the Ampehre framework, which comprises three layers in user space: an extended PAPI library, the Ampehre library, and the Ampehre tools. We base the Ampehre framework on *PAPI*, which makes it inherently portable to other systems running a Linux OS distribution, and we have extended the *PAPI library* to support not only CPU and system-wide sensors but also to retrieve performance data gathered at the accelerator components GPU and FPGA.

Figure 44 denotes the main PAPI components with their interfaces utilized by Ampehre to obtain measurements from the heterogeneous computing resources and the main board of our server node: The PAPI component `rapl` supports CPU measurements, including the cores, last-level cache, memory controller and DRAMs. Modern Intel CPUs provide several so-called Model Specific Registers (MSR) to retrieve data related to energy consumption, temperature, etc. The PAPI component `ipmi` is necessary to retrieve system-wide measurements such as the system-wide power dissipation measured at the power supply. For this, the component communicates with the *Baseboard Management Controller* (BMC) by means of the Linux OpenIPMI library. IMPI is a standard to unify server platform management. The *Nvidia GPU* is supported if PAPI is compiled with the `nvml` component. This component includes the Nvidia Management Library (NVML), which is used to obtain the current power dissipation and die temperature. Finally, Am-

pehre is enabled to gather measurement data on the *Maxeler Vectis* by linking against the MaxelerOS library if the `maxeler` component is enabled in PAPI. From the overall four described PAPI components, we have implemented `maxeler` and `ipmi` from scratch and extended `rapl` and `nvml` in order to support the sensors of interest on our heterogeneous compute node. The node employs a *Dell PowerEdge T620* with two *Intel Xeon E5-2609 v2* CPUs as host processors running CentOS 6.8 Linux with kernel v2.6.32, a PCIe-connected *Nvidia Tesla K20c* GPGPU based on the Kepler microarchitecture, and a PCIe-connected *Maxeler Vectis* FPGA board based on Xilinx Virtex 6 (xc6vsx475t).
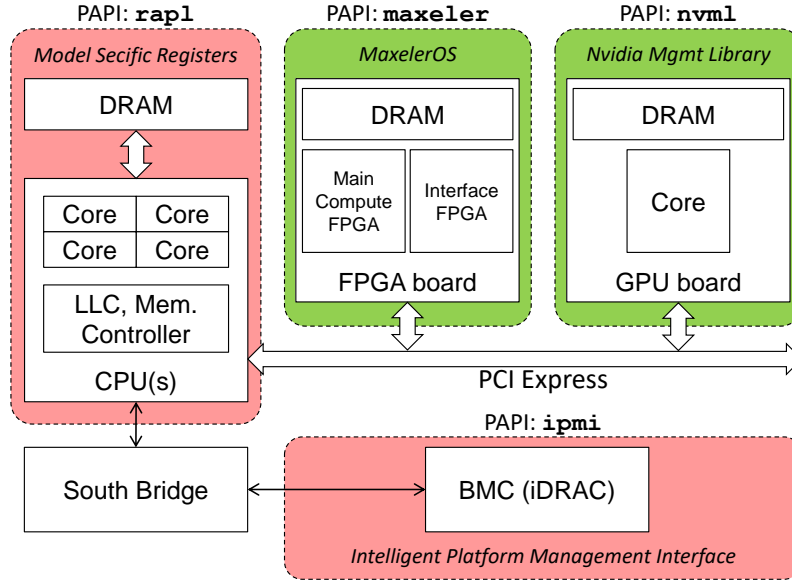
Figure 44: *PAPI components required to retrieve energy and temperature measurements (taken from [LWP18]). We use Linux OS kernel interfaces to sample CPU and BMC sensors (red blocks), and vendor libraries to retrieve measurements from the FPGA and GPU boards (green blocks).*

The *Ampehre library* extends PAPI functionality with the goal to hide all computations and data interpretations from the application developer. The Ampehre library unifies the meaning of gained data across resource boundaries and provides the developer with a set of functions having the same semantics for all resource types. Table 1 gives an overview of the metrics that can be reported by the Ampehre framework for each of the four PAPI components. The measured energy is by definition a value accumulated over the measurement period. For the other quantities, which are power, temperature, utilization, frequency, and amount of allocated memory, Ampehre reports the current (latest) value and the minimum, maximum, and average over the measurement period.

Developers can instantiate the Ampehre library in their applications to use our measurement framework, or they can use one of the following *Ampehre tools*:

hettime extends the well-known Linux utility *time* by reporting comprehensive measurements for an executed binary, i.e., also the energy consumed by the overall system, the average power dissipation and maximum temperature for each component, etc. The results can be stored in JSON files, CSV tables, or simply printed to the shell. `hettime` is highly configurable through command line parameters.

Component	Energy *Accumulated*	Power	Temp.	Utilization	Frequency	Alloc. Memory
			Current, Minimum, Average, Maximum			
`rapl`	✓	✓	✓	✓	✓	✓
`nvml`	✓	✓	✓	✓	✓	✓
`maxeler`	✓	✓	✓	✓	✗	✗
`ipmi`	✓	✓	✓	✗	✗	✗

Table 1: *Quantities that can be measured or computed with Ampehre (taken from [LWP18]).*

msmonitor is a Qt-based live monitoring tool plotting the most recent measurements. `msmonitor` can display the measurement data in the form of an array of curves or as heat maps. These features are exemplary illustrated in Figure 45. The screenshot displays data taken while an arbitrary set of 15 tasks is concurrently executed on CPU, GPU, and FPGA. The array of curves on the left side of Figure 45 represent the current power dissipation of the three computing resources, while the heat maps on the right side of Figure 45 show device utilizations.

Figure 45: *Power dissipation and utilization plotted by* `msmonitor` *while an arbitrary set of 15 tasks are executed on CPU, GPU, and FPGA (taken from [LWP18]).*

msmonitor_cs is a server-client implementation of `msmonitor` for reducing probing effects on the measured server by transferring the GUI rendering to a client connected via TCP/IP.

5   Transparent Acceleration for Heterogeneous Platforms with Compilation to OpenCL

Hardware accelerators, such as GPUs or FPGAs, can offer exceptional performance and energy advantages compared to CPU-only systems. Services that use accelerators are especially interesting in an on-the-fly scenario because they offer higher degrees of freedom in the configuration process, can result in different quality of service aspects,

and finally improve the overall execution. Service providers, however, need to spend considerable efforts on application acceleration without knowing how sustainable the employed programming models, languages and tools are. To tackle this challenge, we developed and demonstrated a new runtime system called HTroP [RVKP19; RVKP18] that is able to automatically generate and execute parallel accelerator code (OpenCL) from sequential CPU code. HTroP transforms suitable data-parallel loops into independent OpenCL-typical work items and offloads the execution of these work items to the hardware accelerators through a mix of library components and application-specific OpenCL host code. Computational hotspots that are likely to profit from parallelization are identified and can be offloaded to different resources (CPU, GPGPU and Xeon Phi) at runtime. We demonstrated the potential of HTroP on a broad set of applications and are able to improve performance and energy efficiency.

OpenCL provides an open standard interface for parallel computing using task- and data-based parallelism, which can be executed across different devices. This means that by generating OpenCL kernel code (once), one can target multiple accelerators. OpenCL, not only poses the challenge of extracting hotspots into kernels and optimizing them for the target accelerator architecture but also involves many tedious adjustments to the remaining host code. Given these challenges, there is a considerable gap between the architectural potential of highly heterogeneous multi-accelerator architectures and their actual adoption and utilization that we aim to overcome with HTroP.

Our approach builds upon and integrates results from different open-source projects: We consider LLVM bitcode as the input format to HTroP, on which all optimization, transformation and acceleration steps are performed. The detection of data-parallel loops is based on LLVM's Polly project [GH16]. Polly uses an abstract mathematical description to detect and model static control flow regions (so-called SCoPs). And finally, we use LLVM's Axtor backend [Mol11] to translate LLVM bitcode into OpenCL kernel code. In our own previous work [DRVP15], we used OpenMP and vectorization to offload hotspots from a low-power client to a remote server with an Intel Xeon PHI accelerator. Related work has researched SCoP-based hotspot detection and acceleration but with other programming models and fewer and different devices in the backend. With Polly-ACC, Grosser et al. [GH16] target Nvidia GPUs using CUDA calls from the host CPU and a PTX backend. Compared to our approach, LLVM bitcode can be generated for a wide range of applications without requiring the source code to be available. Additionally, by using OpenCL as kernel code, various services can be generated on-the-fly, targeting a range of accelerators.

Figure 46 gives an overview of our approach. Our tool flow receives the legacy application in LLVM bitcode and detects computational hotspots as SCoPs. These get parallelized and offloaded using three subsequent optimization passes. In the first transformation step, the *Work-item Parallelizer* uses the dependence analysis information (from Polly) to determine how a loop can be transformed to expose parallelism suitable for OpenCL. For example, Listing 10.1 shows a simple 2D convolution in pseudo code. The outer two **for** loops (line 2 and 4) iterate over the entire input `in`. The inner two **for** loops (line 7-8) perform the convolution for each entry. The dependence info reveals that the innermost loops are data dependent. Hence, only the outer two loops are parallelized.

```
1  heavyConv2D(int *in, int *out, int rows, int cols) {
2    for (int r = 0; r < rows; r++) {
3    // AFTER Work-item Parallelizer for-loop replaced by: int r = get_global_id(0);
```
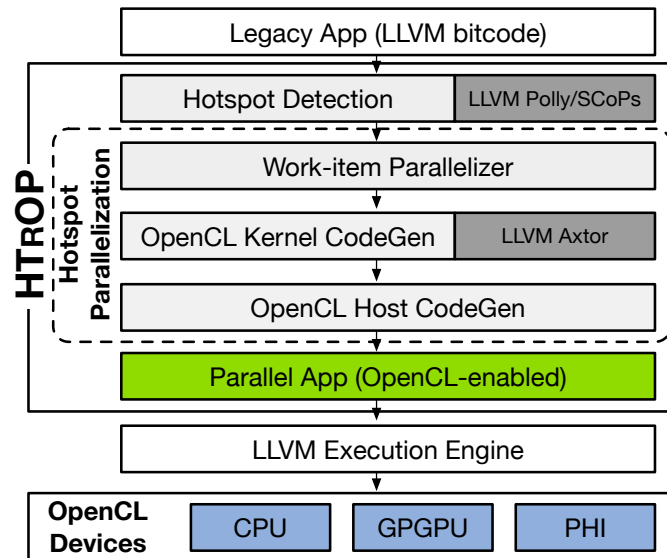
Figure 46: *Architecture of the runtime system. The sequential application is analyzed and parallel OpenCL code is generated on-the-fly.*

```
4       for (int c = 0; c < cols; c++) {
5       // AFTER Work-item Parallelizer for-loop replaced by: int c = get_global_id(1);
6         int |sum| = 0;
7         for (int i = 0; i < 5; i++) {
8           for (int j = 0; j < 5; j++) {
9             // ...
10            |sum| += in[r + i][c + j] * COEFFS[i][j];
11          // ...
12          out[r][c] = |sum|;
```

Listing 10.1: *Nested loops performing a 2D convolution. The two highlighted lines show the modifications performed by the* Work-item Parallelizer *to expose parallelism.*

The following steps are performed to expose work-item parallelism in each loop that has no dependencies:

1. Determine the loop induction variable.

2. Remove the loop control flow.

3. Replace the induction variable with a call to the `get_global_id` OpenCL API call.

The induction variable of a loop represents the variable that is incremented/decremented for each iteration (e.g., `r` and `c` in Listing 10.1). The induction variable can be obtained from the loop header. Once the induction variable is found, we find the corresponding compare instruction that checks the loop exit condition. The compare and branch instructions associated with the loop control flow are removed. This effectively removes the loop structure with all the code previously inside the loop being executed exactly once. The final step is to replace the induction variable with a call to the `get_global_id` OpenCL API call. The lines without line numbers in Listing 10.1 replace Lines 2 and 4 (with Lines 3 and 5) after the *Work-item Parallelizer* is done.

In the second optimization pass, this modified LLVM bitcode is fed into the Axtor-based *OpenCL Kernel CodeGen* to produce corresponding OpenCL kernel code. Since the legacy
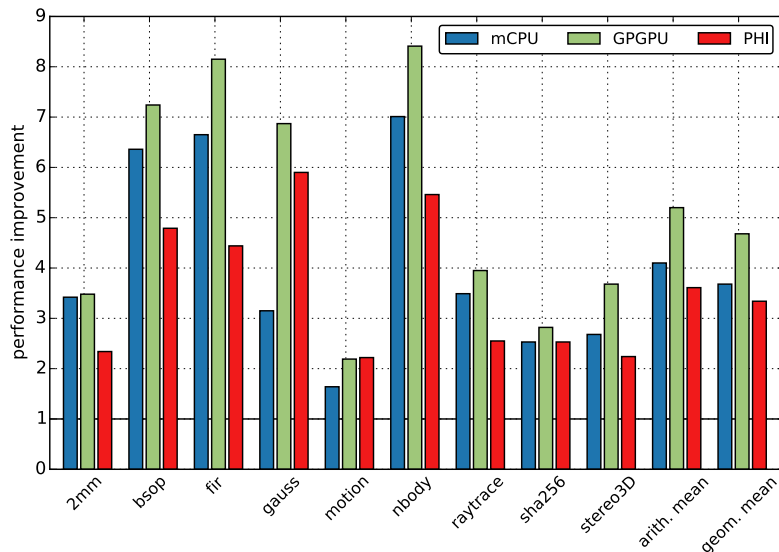
Figure 47: *Performance (speedup) of our runtime system (including all overheads) compared to the normalized CPU baseline (= 1).*

application does not originally support OpenCL, the *OpenCL Host CodeGen* updates the application to support all the devices along with the corresponding OpenCL host code to invoke the kernel. We have implemented a wrapper library that creates and exposes device handles for all appropriate OpenCL devices of our evaluation platform to the global scope of the application. The result is an OpenCL-enabled parallel application that is executed through the LLVM Execution Engine and can offload hotspots to the appropriate OpenCL device.

In order to evaluate our approach, we used the multi-accelerator that we have described in Section 4 and Figure 43. We use a set of benchmark applications extracted from scientific computing, financial, signal- and image processing, and security domains. The baseline is single-threaded CPU code compiled with `gcc` v4.8.2 using the highest optimization level `-O3`. The performance evaluation in Figure 47 reveals speedups for all measured applications with considerable differences between applications and with visible, but small differences among the target devices.

OpenCL turned out to be an effective vehicle for targeting multiple architectures, allowing us to generate the mechanical parts of the host code and to use the same parallelism pattern for the transformation of computationally intensive regions of the application into accelerator code. Service providers can use HTroP in order to generate different variants of services or optimize the execution of services on-the-fly.

6   Conclusion and Outlook

Over the three funding periods of CRC 901, the topic of the use of heterogeneous computing resources in data centers has developed strongly not only in research but especially in practical, economic applications. The ongoing shift of computation from end-user devices to cloud data centers opens up cloud resource providers to leverage heterogeneous compute

resources and benefit from their advantages while keeping the programming interfaces unchanged for service users. This enables faster technological innovation at the hardware level without requiring radical changes in programming models and tooflows on the user side. While the tools presented in this chapter have not been directly taken up on a large scale, the concepts and methods have certainly found their way into practice. For example, FPGA-based overlay architectures are used in Microsoft Bing to implement scoring methods on search results. Heterogeneous programming models with support for CPUs, GPUs and FPGAs as well as runtime systems for the dynamic allocation of tasks to resources are also in widespread use today, e.g. in the SYCL standard which is the basis for Intel's development environments under the name oneAPI.

Last but not least, the extensive experience with FPGA accelerators, programming models and runtime systems has also been incorporated into the design of the FPGA partition of the Noctua 1 and Noctua 2 supercomputers at the Paderborn Center for Parallel Computing. A unique platform has been created that provides a stable production environment for the use of FPGAs in HPC and data-center applications. At the same time, the partition is an ideal testbed for testing communication mechanisms in multi-FPGA applications due to a worldwide unique architecture with an optical L1 network switch, which was developed at the CRC. Thus, a basis for the continuation of this research line exists far beyond the end of CRC 901.

We are pleased to note that the topic examined in CRC 901 has not become stale, even after 12 years of funding. Quite the contrary: although the advantages of highly specialized domain-specific architectures are generally recognized, no other architectures have yet been able to establish themselves apart from GPUs in the data center. One reason for this is certainly that generating efficient code for specialized architectures from abstract specifications remains a major challenge. To have the potential of Domain-Specific Computing, therefore, new approaches are necessary. We also see a high potential for research with impact for methods that globally optimize the operation of a data center, acting across layers. The current approach of increasingly dynamic but still local optimizations of the operating state does not lead to globally optimal operating states. Especially in times of increased volatility of energy price, energy availability, and load from user requirements, feasible methods are needed to cope with the complexity of systems and requirements.

Bibliography

[DRVP15]   Damschen, M.; Riebler, H.; Vaz, G.; Plessl, C.: Transparent offloading of computational hotspots from binary code to Xeon Phi. In: *Proc. Design, Automation and Test in Europe Conf. (DATE)*. EDA Consortium, Mar. 2015, pp. 1078–1083

[GH16]   Grosser, T.; Hoefler, T.: Polly-ACC Transparent compilation to heterogeneous hardware. In: *Proceedings of the 2016 International Conference on Supercomputing*. ACM. 2016, p. 1

[Han21]   Hansmeier, T.: Self-aware Operation of Heterogeneous Compute Nodes using the Learning Classifier System XCS. In: *HEART '21: Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*. Association for Computing Machinery (ACM), 2021

[HBP22]   HANSMEIER, T.; BREDE, M.; PLATZNER, M.: XCS on Embedded Systems: An Analysis of Execution Profiles and Accelerated Classifier Deletion. In: *GECCO '22: Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery (ACM), 2022, pp. 2071–2079

[HKP20a]   HANSMEIER, T.; KAUFMANN, P.; PLATZNER, M.: An Adaption Mechanism for the Error Threshold of XCSF. In: *GECCO '20: Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery (ACM), 2020, pp. 1756–1764

[HKP20b]   HANSMEIER, T.; KAUFMANN, P.; PLATZNER, M.: Enabling XCSF to Cope with Dynamic Environments via an Adaptive Error Threshold. In: *GECCO '20: Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery (ACM), 2020, pp. 125–126

[HP21]   HANSMEIER, T.; PLATZNER, M.: An Experimental Comparison of Explore/Exploit Strategies for the Learning Classifier System XCS. In: *GECCO '21: Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery (ACM), 2021, pp. 1639–1647

[HP22]   HANSMEIER, T.; PLATZNER, M.: Integrating Safety Guarantees into the Learning Classifier System XCS. In: *Applications of Evolutionary Computation, EvoApplications 2022, Proceedings*. Vol. 13224. Lecture Notes in Computer Science. Springer International Publishing, 2022, pp. 386–401

[JM14]   JIN, M.; MARUYAMA, T.: Fast and Accurate Stereo Vision System on FPGA. In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7 (Feb. 2014), no. 1, 3:1–3:24.

[Ken16]   KENTER, T.: Reconfigurable Accelerators in the World of General-Purpose Computing. PhD thesis. Paderborn University, 2016.

[KSP15]   KENTER, T.; SCHMITZ, H.; PLESSL, C.: Exploring Trade-Offs between Specialized Dataflow Kernels and a Reusable Overlay in a Stereo Matching Case Study. In: *Int. Journal of Reconfigurable Computing (IJRC)* (2015), pp. 1–24

[LKEW]   LÖSCH, A.; KNORR, C.; EL-ALI, A.; WIENS, A.: *Ampehre: Accurately Measuring Power and Energy for Heterogeneous Resource Environments*. http://ampehre.uni-paderborn.de/. Last accessed on Jan 3, 2023

[LP17]   LÖSCH, A.; PLATZNER, M.: reMinMin: A Novel Static Energy-Centric List Scheduling Approach Based on Real Measurements. In: *Proceedings of the 28th Annual IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 2017

[LP18]   LÖSCH, A.; PLATZNER, M.: A Highly Accurate Energy Model for Task Execution on Heterogeneous Compute Nodes. In: *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2018

[LP20]   LÖSCH, A.; PLATZNER, M.: MigHEFT: DAG-based Scheduling of Migratable Tasks on Heterogeneous Compute Nodes. In: *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2020

[LWP18]   LÖSCH, A.; WIENS, A.; PLATZNER, M.: Ampehre: An Open Source Measurement Framework for Heterogeneous Compute Nodes. In: *Proceedings of the International Conference on Architecture of Computing Systems (ARCS)*. Vol. 10793. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 73–84

[MKP20]   MEYER, M.; KENTER, T.; PLESSL, C.: Evaluating FPGA Accelerator Performance with a Parameterized OpenCL Adaptation of Selected Benchmarks of the HPCChallenge Benchmark Suite. In: *2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)*. IEEE. 2020, pp. 10–18

[MKP22]   MEYER, M.; KENTER, T.; PLESSL, C.: In-depth FPGA accelerator performance evaluation
           with single node benchmarks from the HPC challenge benchmark suite for Intel and Xilinx
           FPGAs using OpenCL. In: *Journal of Parallel and Distributed Computing* 160 (2022),
           pp. 79–89.

[MKP23]   MEYER, M.; KENTER, T.; PLESSL, C.: Multi-FPGA Designs and Scaling of HPC Challenge
           Benchmarks via MPI and Circuit-Switched Inter-FPGA Networks. In: (2023).

[Mol11]   MOLL, S.: Decompilation of LLVM IR. In: *Master's thesis* (2011)

[MSZ⁺11]   MEI, X.; SUN, X.; ZHOU, M.; JIAO, S.; WANG, H.; ZHANG, X.: On Building an Accurate
           Stereo Matching System on Graphics Hardware. In: *Proc. ICCV Workshop on GPU in
           Computer Vision Applications (GPUCV)*. IEEE, 2011

[RVKP18]   RIEBLER, H.; VAZ, G.; KENTER, T.; PLESSL, C.: POSTER: Automated Code Acceleration
           Targeting Heterogeneous OpenCL Devices. In: (2018)

[RVKP19]   RIEBLER, H.; VAZ, G.; KENTER, T.; PLESSL, C.: Transparent Acceleration for Heterogeneous
           Platforms With Compilation to OpenCL. In: *ACM Transactions on Architecture and Code
           Optimization (TACO)* 16 (2019), no. 2, pp. 1–26

[SHW⁺14]   SHAN, Y.; HAO, Y.; WANG, W.; WANG, Y.; CHEN, X.; YANG, H.; LUK, W.: Hardware Accelera-
           tion for an Accurate Stereo Vision System Using Mini-Census Adaptive Support Region.
           In: *ACM Transactions on Embedded Computing Systems (TECS)* 13 (Apr. 2014), no. 4s,
           132:1–132:24

[TLLA14]   TIPPETTS, B.; LEE, D. J.; LILLYWHITE, K.; ARCHIBALD, J. K.: Hardware-Efficient Design of
           Real-Time Profile Shape Matching Stereo Vision Algorithm on FPGA. In: *Int. Journal of
           Reconfigurable Computing (IJRC)* (2014)

Subproject C4:

On-The-Fly Compute Centers II: Execution of Composed Services in Configurable Compute Centers

Holger Karl[3], Marten Maack[2], Friedhelm Meyer auf der Heide[2], Simon Pukrop[2], Adrian Redder[1]

1 Department of Computer Science, Paderborn University, Paderborn, Germany

2 Heinz Nixdorf Institute and Department of Computer Science, Paderborn University, Paderborn, Germany

3 Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

1 Introduction

OTF compute centers are intended to exploit and support the characteristics of OTF services. They also are expected to exist at various scale, ranging from full-fledged data centers down to edge-computing semi-racks or even smaller. A characteristic of OTF services is that they are composed of components with explicit, quantitative meta data about those compositions – e.g., resource consumption per component, data flows between components, etc. OTF centers should therefore exploit this meta data to improve service performance and system efficiency. Moreover, OTF centers can be typically highly heterogeneous, having various types of computation units and persistent storage units. If a service provides metadata about its performance on different types of computation units, such information is also used to make better scheduling decisions as well. OTF centers also have one or more networks that connect these resources with each other. An OTF service can be provided by a single or several cooperating (sometimes also competing), geographically or organizationally distributed OTF compute centers. If necessary, they are supplemented by resources temporarily rented from the cloud.

The OTF services to be executed are usually composed of several interlinked, interacting components. Ideally, information about resource consumption, such as runtime and memory, is available for these components, possibly for different computing units such as CPUs, GPUs, and FPGAs. Information about the interaction of these components is available, such as the amount of data to be exchanged and minimum data rate requirements.

We have focused on resource management within as well as between data centers. We have worked from abstract, algorithmic models to very concrete framework-specific aspects, with methodological lines ranging from approximation and online algorithms with provable quality guarantees to system design and evaluation platforms. In doing so, we

holger.karl@hpi.de (Holger Karl), marten.maack@hni.upb.de (Marten Maack), fmadh@upb.de (Friedhelm Meyer auf der Heide), simonjp@hni.upb.de (Simon Pukrop), aredder@mail.upb.de (Adrian Redder)

have considered not only classical efficiency metrics such as throughput or utilization, but also energy consumption, for example.

In **Section 2.1:** *Approximation Algorithms for Scheduling*, we deal with the complexity of variants of scheduling problems. For this, we introduce new, and extend known models that capture important properties and features such as energy efficiency, the problems arising when global resources are available, the impact of setup times in reconfigurable systems, and the challenges arising when the compute center may delegate parts of the work to clouds. In these theoretical investigations, we have concentrated on algorithmic and complexity-theoretic approaches. On the one side, we have proven hardness results; on the other, we have developed approximation algorithms and proven bounds on their approximation quality.

In **Section 2.2:** *Distributed Execution of Service Chains*, we present extensions of formal description techniques towards OTF services. Further model extensions describe heterogeneous but interchangeable resources (e.g., CPU vs. FPGA). Based on these models, we have developed algorithms and mechanisms: which resources (data rate, compute capacity, . . .) are allocated to which component to which task on which server. We have considered algorithms for both offline and online variants and have evaluated them both experimentally and theoretically. Hand in hand with the experimental analyses, we have also used the properties of real input streams (actual traces) as a starting point for modeling such streams in order to perform more realistic experimental analyses. In addition, we have developed heuristic or approximate solutions for these resource management problems and applied experiments and competitive analysis to evaluate their quality, partly based on realistic workloads.

2   Main Contributions

We structure the description of the main contributions of our subproject into the above-mentioned two sections.

2.1   Approximation Algorithms for Scheduling

The area of scheduling generally deals with the planned processing of tasks. From a computation perspective, addressing this topic leads to optimization problems that are typically combinatorial in nature and—in all but the simplest cases—NP-hard.[17] Hence, even if the complete instance of such a problem is known, there is little hope for an efficient algorithm that is guaranteed to find an optimal solution. One way of approaching this problem is to design algorithms that guarantee a certain quality in the produced solutions. In particular, an α-approximation for an optimization problem is guaranteed to produce a solution with an objective value that is within a factor of α of the optimum. The parameter α is called the *approximation ratio* or *guarantee* and, if not stated otherwise, the term *approximation algorithm* is used for algorithms that have a running time bounded by a polynomial in the input length of the problem. One of the earliest works in this direction was done by Graham [Gra66] in the 1960s regarding a fundamental scheduling problem.

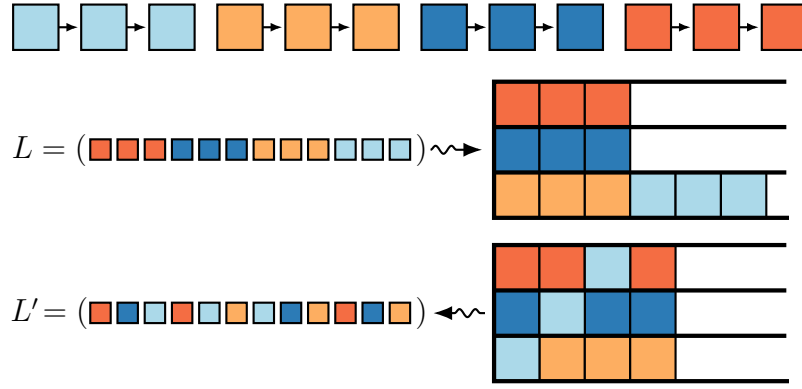[17]They cannot be solved efficiently if P ≠ NP, which is generally assumed to be true.

Figure 48: *A simple example for a scheduling problem with 12 unit time jobs with precedence constraints and three machines. For the first provided list L the list scheduling algorithm yields a schedule with makespan six while 4 is optimal. Generalizing this example, ratios arbitrarily close to 2 may occur. The list scheduling algorithm would have found an optimal schedule given list L'.*

We briefly discuss this classical result to make the topic more tangible and to introduce some of the basic concepts.

In the respective problem, a given set of jobs has to be assigned to a set of identical machines. Each job j has a processing time p_j, and between any pair of jobs (j, j') there may be a precedence constraint $j \prec j'$, that is, job j' can be processed only after job j is completed in this case. Furthermore, once the processing of a job is started, it cannot be interrupted (no preemptions), and the objective is to minimize the point in time in which the last job is completed—the *makespan*. Graham [Gra66] introduced the *list scheduling* algorithm for this problem, which arranges the jobs in a list and always schedules the first job on the list for which all precedence constraints are satisfied at the next possible time that is as soon as there is an idle machine. In this work, it was shown that list scheduling is a 2-approximation by considering two cases: Either all machines are working, or there are idle machines. In the first case, the algorithm behaves optimally, and in the second, either there are no more jobs or all the remaining jobs depend on the ones that are being executed. The latter observation can be used to bound the times with idle machines against the length of the longest chain of succeeding jobs which, in turn, is a lower bound for the optimum. Hence, both the times with and without idle machines can be upper bounded by the optimum, yielding the proof that list scheduling is a 2-approximation.[18]
In Figure 48, an example is provided showing that the analysis cannot be substantially improved. Essentially, the best we can hope for regarding approximation algorithms for NP-hard problems are so-called approximation schemes: A *polynomial time approximation scheme (PTAS)* is a family of approximation algorithms (with polynomial running time) that provide a $(1 + \varepsilon)$-approximation for each $\varepsilon > 0$. Moreover, if the running time of the scheme is bounded by a polynomial in both the input length and $1/\varepsilon$, it is called *fully polynomial (FPTAS)*.

Interestingly, the list scheduling algorithm essentially still works in an *online* setting where the jobs are revealed over time during the processing time (they can be appended to the list). It is easy to see that in such an online algorithm, no algorithm can be guaranteed

---

[18]The actual analysis is slightly sharper.

to find an optimal solution for each input instance. However, there is an established way to measure the quality of an online algorithm that is closely related to the concept of approximation algorithms. In particular, it is considered *c*-competitive if the objective value achieved by the algorithm is guaranteed to be within a factor of *c* of an optimal *offline* solution.[19] Note that since the algorithm cannot know when the instance is completed, it has to *maintain* the above property for the respective instance seen so far. Coming back to the list scheduling algorithm as an example, it is easy to see that it barely uses any information about the instance, which is why the mentioned analysis can be adapted to show that it is 2-competitive.

Since the 1960s, the study of scheduling problems has expanded massively both in breadth and depth. For a broad overview, we refer to the textbook by Pinedo [Pin16]. In the context of the present subproject, however, approximation algorithms and to a much smaller extent, online algorithms have primarily been considered for areas of scheduling with particular relevance to OTF computing. In the following, we discuss the most prominent of the considered directions and highlight some of the most important results achieved in the subproject. Here we put the strongest focus on the topic we have dealt with the most at the end of the project, that is, cloud assisted scheduling.

2.1.1 Energy-Efficient Scheduling

In the study of scheduling, there is typically a strong focus on optimizing performance. Indeed, probably the most studied objective function in scheduling is the makespan, that is, the point in time the last task of an instance is completed. In many contexts, however, performance is neither the only nor the most important factor to consider, and one additional aspect that is of particular importance in today's world is energy efficiency. It is not quite obvious how to best capture energy consumption in a theoretical model, but a very influential approach to do so was introduced in 1995 by Yao et al. [YDS95] in a seminal work: In the *speed-scaling model*, the clock rates of processors can be changed at runtime, with slower clock rates resulting in lower overall energy consumption for the computation. A crucial factor is that the energy consumption grows superlinearly with the clock rate, with experiments pointing to growth with the third to fifth power in the clock rate for some real-world settings [BBS+00]. Moreover, the model relates to real-world techniques such as AMD's *PowerNow!* or Intel's *SpeedStep*.

As hinted above, a typical function modeling the energy consumption is of the form of s^{α}, where s is the clock speed of the processor, and α is a constant, usually between 3 and 5. In the classical work by Yao et al. [YDS95], a set of given jobs with different release dates, deadlines, and workloads has to be scheduled preemptively—the processing of a job may be interrupted, and resumed at a later time—on a single speed-scalable processor. The goal is to finish all the jobs in an energy-minimal way and both offline and online approaches are provided to that end. In a later work due to Chan et al. [CLL11], jobs are additionally associated with values and it is no more required to finish all jobs before their deadline but rather a combined objective of spend energy and lost profit is considered.

In [KP13], we generalize and improve upon the work by Chan et al. [CLL11]. We

---

[19]Depending on the problem, the concept is sometimes defined slightly differently, that is, up to additive constants.

consider the online setting and develop a combinatorial greedy algorithm that guarantees a competitive factor of α^α, which is optimal at least for greedy algorithms. In [CLL11], on the other hand, a $(\alpha^\alpha + 2e\alpha)$-competitive algorithm was presented for the single processor case. Moreover, the analysis of the algorithm uses techniques that are significantly different from the typical potential function argument. We utilize well-known tools from convex optimization and duality theory, in particular those that have already proved useful in the original work by Yao et al. [YDS95]. The developed algorithm, in some sense, can be seen as a combination of similar convex programming techniques with a carefully crafted greedy approach.

In [ABC$^+$17], we consider a relaxed version of one of the central problems in speed-scaling: scheduling with respect to a combined objective of energy consumption and response time. While the problem regarding unit-sized jobs was well understood before, our results explore two important additional aspects, namely, arbitrary job sizes and discrete speed levels, which arguably model actual technology more accurately. Our results in [ABC$^+$17] represent the first step in several years to solve the complexity question of this problem. More precisely, for the relaxation with *fractional* response times, we provide an efficient and optimal algorithm that follows a geometric approach utilizing certain structural properties that are obtained from an integer linear program and its dual.

2.1.2 Scheduling with Global Resources

In many real-world scheduling scenarios, different machines are connected via additional shared resources. Early considerations in this direction have already been made in the 1970s by Garey and Graham [GG75] building on the seminal work due to Graham [Gra66], discussed above. However, in the scheduling literature processors are very often assumed to be independent of each other. In contrast, we have considered models in which m identical machine share one additional resource, corresponding to, for instance, the data rate of a memory bus connecting processors being limited. The tasks to be processed are described by their processing times and resource requirement. The scheduler distributes tasks to processors and manages the access of the processors to the shared resource. If a task receives only a fraction of its resource requirement in a time step, its execution is slowed down accordingly. For example, a job of size p can be processed in p time units if it receives its full resource requirement in each time step. If it receives only half of its request in each time step, the processing time increases to $2p$. The goal is to minimize the makespan. Our key results regarding this scheduling problem are presented in [KMRS17]. We show the problem to be NP-hard and provide an efficient approximation algorithm with an approximation ratio of $2 + 1/(m-2)$. The algorithm utilizes a sliding windows approach that considers jobs ordered by non-decreasing resource requirement and, for each time step, tries to find a subset of consecutive jobs such that all but one can be completed using the full resource. Furthermore, we consider a variation of this model involving composed tasks consisting of multiple components, each of which has its own resource requirement. A task is completed when all of its components are completed and the goal is to minimize the average completion time of all task. We again show the problem to be NP-hard and provide an approximation algorithm with a ratio $2 + 4/(m-3)$ up to an additive constant.

2.1.3 Scheduling with Setup Times

In reconfigurable systems such as systems of FPGAs, a considerable amount of hardware configuration may be required when switching between tasks of different types. For scheduling in such systems, we have investigated a straightforward model in which we consider n tasks divided into k classes on a set of m processors. The processors have to be reconfigured to process the different classes. That is, each time a batch of jobs from a fixed class is to be processed on a processor a (possibly class dependent) setup time has to be paid. In [MMMR15], we provided the first results regarding this model with identical machines, including a $(3/2 + \varepsilon)$-approximation and an FPTAS for the case with a constant number of machines. This first result has quickly inspired further investigations from other researchers. For instance, Jansen and Land [JL16] provided a very simple and fast 3-approximation as well as a PTAS for the problem. In a follow-up work [JMM19], we considered generalized machine models: First, we developed a PTAS for the case with uniformly related machines, where the processing time of a job (and the setup time of a class) is scaled according to a machine-dependent speed factor. In the case of unrelated processing times (and setup times), we showed that no approximation algorithm with ratio $\Omega(\log n \log m)$ is possible unless a common hypothesis from complexity theory fails. We also provided a randomized algorithm with a matching upper bound. Lastly, we considered variants on identical machines with assignment restrictions and provided both hardness results and constant factor approximation algorithms.

2.1.4 Cloud Assisted Scheduling

Nowadays, a big part of web traffic and computational tasks are handled by large cloud providers such as Amazon Web Services and Microsoft Azure. Naturally concluding from that, a part of the CRC considered a setting in which computational resources are rented from cloud providers, exclusively or additionally. We present two different approaches, one where all jobs must be scheduled in the cloud and another where we own some free hardware that can be enhanced by rented cloud machines.

Cost-efficient scheduling on machines from the cloud: We consider the former approach in [MMMR18]. In that model, an online scheduler has to rent machines of a certain type for some arbitrary duration to ensure that all jobs can be scheduled before their respective deadline. Additionally, there is some machine-type dependent setup time s, before a newly rented machine can be used. The goal is naturally to minimize the cost paid to rent the machines. To be more specific, we assume that there are exactly two different machine-types, which differ in their price and their setup time. Jobs, on the other hand, consist of some processing time per machine-type, a release date, and a deadline. A critical parameter in this paper is the minimum slack β, which is the minimum time between any job release and the latest point where that job has to be scheduled to hit its deadline. Our paper has two main results: First, if the setups are large in comparison to the minimum slack ($s > \beta$) no finite competitiveness is possible. Secondly, if $\beta = (1 + \varepsilon)s$, for some ε with $1/s \leq \varepsilon \leq 1$, we give an algorithm that only depends on ε and the ratio of machine prices, and is proven to be optimal up to a factor of $O(1/\varepsilon^2)$.

Server cloud scheduling: In [MMP21] we both incorporate the possibility to allow some already owned hardware that can be augmented via the cloud, as well as imagining a big

task that can be represented as a graph of small jobs that depend on each other. This later part of our research combines various properties from different scheduling models, of which most have already been studied individually. Those are, in no particular order, unrelated machines, cost minimization for rentable machines, precedence constraints between jobs, and communication delays between the types of machines. We try to present this model in a bit more detail and describe one of the two main results.

We consider a scheduling problem SCS in which a task graph $G = (\mathcal{J}, E)$ has to be scheduled on a combination of a local machine (server) and a limitless number of remote machines (cloud). The task graph is a directed, acyclic graph. Each job $j \in \mathcal{J}$ has a processing time on the server $p_s(j)$ and on the cloud $p_c(j)$. The values of p_s and p_c can be arbitrary in $\mathbb{N}_0$, meaning that the server and the cloud are unrelated machines. An edge $e = (i, j)$ denotes precedence, i.e., job i has to be fully processed before job j can start. Furthermore, an edge $e = (i, j)$ has a communication delay of $c(i, j) \in \mathbb{N}_0$, which means that after job i finished, j has to wait for an additional $c(i, j)$ time steps before it can start, if i and j are not both scheduled on the same type of machine (server or cloud). A schedule π is a partition of the jobs into two sets: jobs processed on the server and the cloud, respectively. Additionally, a schedule assigns some starting time to each job. The cost ($cost$) of the schedule is then the total processing time of jobs processed on the cloud, and the makespan ($mspan$) is the completion time of the last job. For a schedule to be feasible, the following conditions must hold:

- Each job only starts after it is available, which means that all predecessors have finished processing and relevant communication delays have passed.

- No two jobs process on the server in parallel.

- If there is a budget, the $cost$ may not exceed it.

- If there is a deadline, the $mspan$ may not exceed it.

Naturally, if there is a budget, the goal is to minimize the deadline. If there is a deadline, the goal is to minimize the cost.

We categorize different sub-problems by their task graph structure and different processing times. The main results are an FPTAS with respect to the makespan objective for a fairly general case and strong hardness for the case with unit processing times and delays.

Imagine a task graph drawn in such a way that every edge goes from left to right. Now assume that we split the jobs in this task graph into a left part ($\mathcal{J}_l$) and a right part ($\mathcal{J}_r$), so that there are edges from $\mathcal{J}_l$ to $\mathcal{J}_r$, but no edges from $\mathcal{J}_r$ to $\mathcal{J}_l$. In other words, in a running schedule, $\mathcal{J}_l$ and $\mathcal{J}_r$ could represent already processed jobs and still be processed jobs, respectively. For any given task graph, we call the maximum number of edges between $\mathcal{J}_l$ and $\mathcal{J}_r$ the *maximum cardinality source and sink dividing cut* of the graph. We discuss how to solve or approximate SCS problems with a constant size cut, but otherwise arbitrary task graphs. We present the deadline-confined cost minimization; in the paper, we also show how to adapt this to the budget-confined makespan minimization. We start by describing a dynamic program to optimally solve instances of SCS with arbitrary task graphs. At first, we will not confine the algorithm to polynomial time. Consider a given problem instance with $G = (\mathcal{J}, E)$, processing times $p_s(j)$ and $p_c(j)$ for each $j \in \mathcal{J}$, communication delays $c(i, j)$ for each $(i, j) \in E$, and a deadline d. We define intermediate states of a (running) schedule as the states of our dynamic program. Such a state contains two types
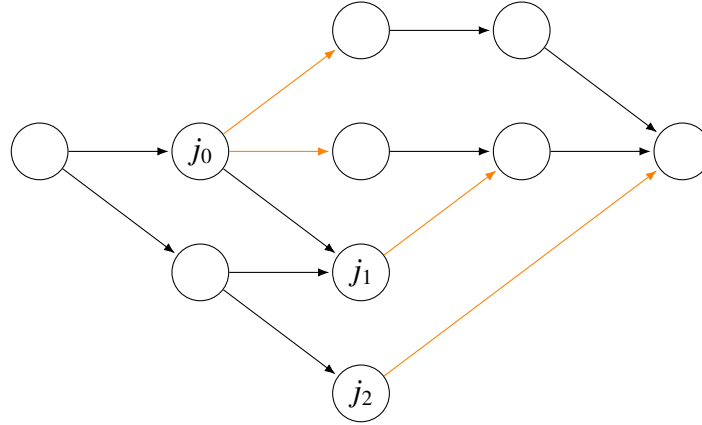
Figure 49: *Example for a small task graph with some state of a schedule. j_0, j_1 and j_2 represent jobs that are already processed but have some unprocessed successor remaining. Open edges are marked orange.*

of variables. First, we have two global variables, how many time steps have passed since the beginning and the number of consecutive time steps the server has been idling (counted from end to start). The second type is defined per *open edge*. An open edge is a $e = (j, k)$ where j has already been processed, but k has not. For each such edge $e = (j, k)$ add the following variables: the edge itself, whether j was processed on the server or the cloud, and the number of time steps passed that passed since j's completion. Note here that we purposefully drop the completion time and location of every processed job without open edges, as those are not important for future decisions anymore. There might be multiple ways to reach a specific state, but we only care about the minimum possible cost to achieve that state, which is the *value* of the state. We iteratively calculate the value of every state reachable in a given time step $= 0, 1, 2, \ldots$. This state forms the beginning of our *state list*. We exhaustively calculate every state that is reachable during a specific time step, given the set of states reachable during the previous time step. Intuitively, we try every possible way to "fill up" the still undefined time windows of the server idling, and time passed since some j of an open edge was completed. After the current time step reaches our deadline, we can select the cheapest option from among the states to get the optimal schedule. This algorithm is polynomial in the deadline, but that can be exponential in the input size. To get an approximation algorithm that is polynomial in the input size, we scale all processing times in relation to the deadline and the input size. While doing so, we can $(1 + \varepsilon)$-approximate the optimal solution in time $poly(n, \varepsilon)$, for any $\varepsilon > 0$, which in turn means that we described an FPTAS for this problem. The other main result of the paper is a proof that the SCS problem is strongly NP-hard, even if all processing times and communication delays are equal to 1.

2.2 Distributed Execution of Service Chains

For the entire duration of the CRC, we have worked on basically the same scenario: the distributed execution of service chains in a complex infrastructural concept. Let us dissemble these terms first before digging into any more detailed contribution descriptions. First, the services we are considering are not monolithic services provided by a

single executable, e.g., a server process running somewhere. In line with developments in software engineering, a monolithic service executable is broken down into smaller independently executable pieces of software. They are connected together to collectively provide a service. The load in such a scenario can widely differ: A service may be invoked once by an individual user or repeatedly; a more interesting case is when a whole user population requests a service for repeated execution, an entire stream of requests arrives. The underlying infrastructure in such cases can be quite diverse: It can run from a tightly controlled data center to an edge-computing scenario in wide-area networks that is still under operational control of a single entity, to services and/or user populations that are spread over many administrative domains with independent control. In all these scenarios, there is a range of typical problems to solve in order to deal with load:

1. How many instances of a particular component service should be executed?

2. Where should these instances be placed?

3. Which request from which user is assigned to which instance; after processing one step in a service chain, requests from which instance are forwarded to which other instance?

These problems are known as the scaling, placement, and routing problems for service chains. In addition, there are further problems to solve, such as state management, deploying executable artefacts, etc. Most of the work described in this section deals with these problems under different perspectives.

In the following subsections, we first describe our contributions to these problems in the context of computing inside a wide-area operator network (In-network computing; Sections 2.2.1, 2.2.2, 2.2.3). Finally, in Section 2.2.4, we consider data center scenarios.

2.2.1 Description Techniques

When trying to deploy a service into a network, it is necessary to understand the characteristics and properties of such a service. From a purely functional perspective, it suffices to think of a service chain as a graph of atomic components, connected in a direct (typically, acyclic) graph with explicitly marked ingress and egress points. During operation, that knowledge suffices to forward one request along the chain (with additional information to which particular instance to forward to).

But during deployment (and reconfiguration), only functional information is insufficient to properly dimension resources. A better understanding of the required resources that a service or its components need is required. More specifically, what is the relationship between, on one hand (a) the load a component has to process (e.g., as a rate of temporal Poisson process) and (b) the resources that are assigned to it (e.g., the number of virtual cores, in a normalized manner) and, on the other hand, the resulting performance of such a component, e.g., the throughput it can sustain or the per-request delay. In early publications in this context, we have derived description formats to express such load-resource-performance profiles in a standardized manner, for individual components, services, and recursively defined services.

Here, we want to emphasize an aspect that has received little consideration and was first investigated in [SSKW19], jointly with colleagues from other CRC projects. The question

occurs what happens if the service graph splits into multiple paths from ingress to egress or if the service is request/reply-style, expecting an answer. Then, it matters whether subreplies from individual paths can be used independently or whether they need to be synchronized. Reference [SSKW19] shows that, all else being equal, this information matters for optimal deployment. That reference also introduced a Petri-net-based formalism to express such synchronization properties. Starting from a modeling formalism, we show that it is possible to automatically generate simulation programs or input files for Petri net solvers to assess the performance of concrete services. Moreover, thus modeled services can be fed into orchestration system that can leverage information about synchronization requirements for better orchestration decisions.

2.2.2 Orchestration

The above section has already mentioned the notion of "orchestration"—it is an umbrella term to capture all decisions that need to be taken when deploying a distributed service into a concrete infrastructure (e.g., as mentioned above, scaling, placement, and routing). In this section, we describe various algorithmic problems that we have talked in the orchestration context.

Conventional Orchestration Approaches A "conventional" approach is an approach that assumes full knowledge about the services to be orchestrated and their constituting performance profile, about the underlying infrastructure, and about the load patterns. This line of work culminated, in a sense, in Reference [DKM18]. The JASPER system proposed therein combined most of the aspects we had considered in previous papers and automatically deals with scaling, placement, and routing. It takes service templates and monitoring data of the underlying infrastructure as input and solves scaling, placement, and routing in an *integrated* optimization process (Figure 50), unlike separated, individual processes that were common in the literature before that. It handles dynamically adding services and services that terminate after completion, taking account of the current resource situation, and uses service templates in line with what was described before in Section 2.2.1. The reference also shows that the considered problem is NP-complete (via a set cover reduction proof). JASPER is also flexible in the way different optimization objectives can be combined and in that constraint violations can be acceptable, but their number is minimized. The solution approach is a mixed-integer linear program, with the typical limitations on problem size and require solution time. These limitations are amended by a heuristic. At the time of publication, a fairly unique feature of the heuristic was its capability to start from an existing solution and look for small modifications to accommodate new services upon arrival (Figure 51). This is considerably faster than always starting from scratch with marginal reductions of solution quality.

As a more specific example of optimization potential, we point out Reference [KK17]. It was one of the first papers to look at optimizing response time for such service chains, conceiving of the entire system as a queuing system where queuing delay is a dominant contributor to delay. The challenge was to find a good comprise for the non-linear time-in-system formula in a queuing system. We tackled this by developing a custom-tailored linear approximation to be used in a linear optimization program.
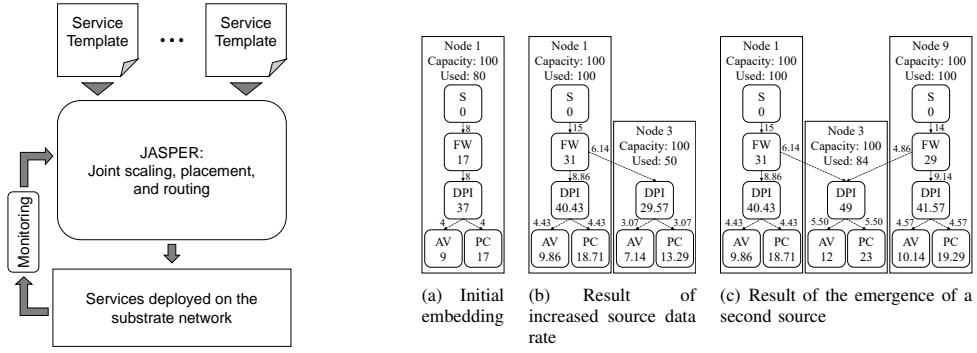
Figure 50: *Main control loop of the inte-grated JASPER approach for orchestration (based on Figure 2 in [DKM18]).*

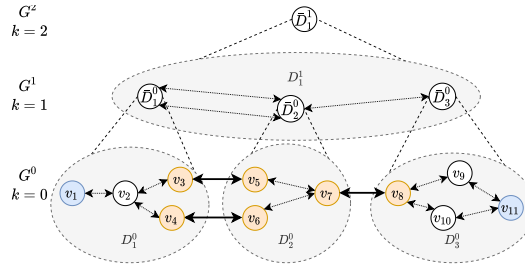Figure 51: *Steps of JASPER when load situation changes (Figure 6 in [DKM18]).*

Figure 52: *Example with k = 2 hierarchies. Ingress and egress nodes are shown in blue, border nodes of a domain in orange (Figure 1 in [SJK21]).*

Hierarchical Orchestration   Despite all improvements we did to the orchestration process, it still stayed a fairly complex problem. It stands to reason to break it down into subproblems. Moreover, in a multi-provider environment, it is unreasonable to assume that competing providers provide information about their infrastructure to each other. Both make a central perspective on the orchestration problem questionable. We hence developed a hierarchical approach to orchestrate services [SJK21] (Figure 52). The challenge was to find a good separation of available information and responsibility. Inspired by well-known multi-provider routing problems (known, e.g., from MPLS PCE contexts), we need to not only account for the data rates, but also for computational capacity. We did so by abstracting the capacity of a domain and only reported aggregated information to the higher level. Recursively, this ensured conservative orchestration choices trading off optimality for scalability.

Distributed Orchestration   An alternative to hierarchical orchestration is to build an entirely distributed orchestration approach where each node works for itself. The challenge here is to deal with the non-locality of the orchestration problem: Resources might be available outside any node's observational horizon that still could result in a better solution. Hence, there is an inherent greediness involved in our distributed approach [SKK20].[20] More specifically, we looked at a locally greedy scheme—which processes

---

[20]It bears mentioning that this paper resulted from a Bachelor thesis.

Service	Component	Node 1	Node 2	...
S1	C1	20%	30%	...
S1	C2	40%	0%	...
S2	C1	30%	60%	...
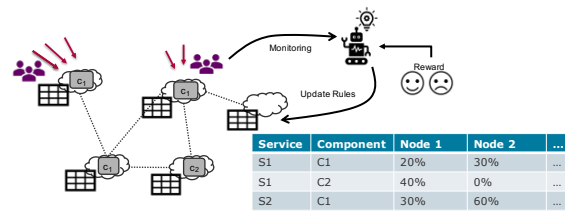
Figure 53: *Service orchestration as a centralized learning problem.*

any stage in a service chain once there is sufficient capacity available at a node—and combine it with an routing scheme where requests are forwarded on a shortest path towards their egress node, hoping that there will be sufficient capacity on the way to process remaining steps in a service chain (we did assume that every node can compute any stage in any chain, i.e., that all deployment units are available everywhere). This request forwarding does adjust to locally observable capacity information, rerouting a request away from already overloaded links. As a consequence, this scheme only needs global structural information (in particular, shortest paths) that change on long time scales and can reasonably be assumed to be available, but it does not assume non-local capacity or utilization information. It turns out that centralized heuristics are (unsurprisingly) still competitive with such distributed schemes but that distributed schemes achieve almost comparable performance at significantly reduced cost.

Machine-Learning-Based Orchestration   All previously described orchestration approaches where "conventional" in the sense that they started from expert knowledge about the problem, the environment, and possible solution approaches. While this lead to interesting results and workable solutions, it is also promising to investigate currently popular machine-learning-based approaches and see how they fit to the orchestration problem. Specifically, reinforcement learning is a natural candidate, with an agent making orchestration decisions and obtaining rewards from the environment, e.g., informing it about how many flows could have been successfully processed or what relevant quality-of-service characteristics (e.g., request latency) were achieved. Typically, challenges to deal with are how to encode a network and services in fixed-length inputs and state representations necessary for an agent, how to encode suitable actions, and how to deal with delayed or sparse rewards or with uncertainty about service or infrastructure descriptions.

One way to deal with the state-size problem is investigated in Reference [SKM+21]. The key idea is, for each node, to use a table with service components as rows, other nodes as columns, and as an entry the probability with which to forward a request for a particular service component from one node to the node in the respective column. These tables are learned by a central agent based on delayed monitoring data and are periodically distributed back into the actual network. An illustratoin of the learning procedure is shown in Figure 53. The actual decisions are all taken locally (only needing a table lookup and a generation of a single random number). The results showed that such a centralized, delayed-observation-delayed-reward approach works surprisingly well, but it is obviously limited in scale. Separate tables need to be trained for each node.

To improve that situation, we compared it to another approach: Instead of training separate tables per node, we trained individual agents per node that had more freedom for decisions
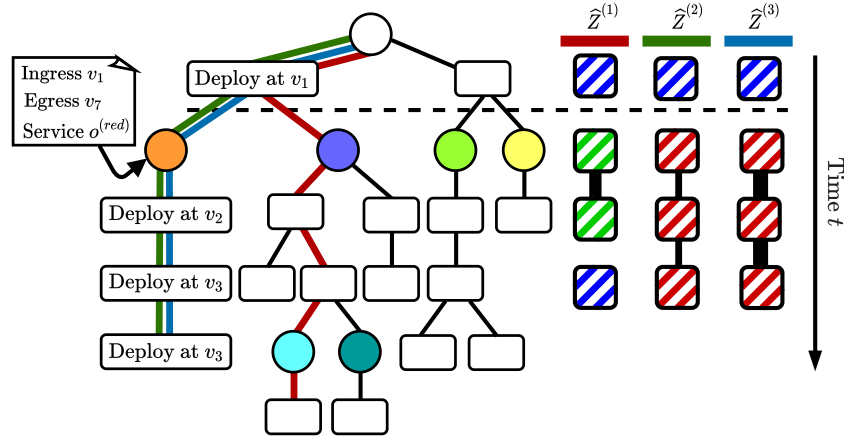
Figure 54: *FutureCoord plans service coordination beyond the current flow (left) with forecasts of future demands (right); Figure 2 of [WSK22].*

[SQK21]. This does improve the scalability of the overall concept in most cases; only in the case of a deadline for service execution is the first approach superior.

Approaches such as these are interesting, but they do entirely disregard any a priori understanding of the problem, at least at learning and inference time they are *model-free*. In addition, they do need expert knowledge to set up the representation of states, observations, and actions as well as to select the right reinforcement learning algorithms and neural network structures. It should make sense to incorporate explicit knowledge into a machine-learning approach, turning it into a *model-based* approach. FutureCoord [WSK22] is such an approach. Unlike many ML approaches, it is based on Monte-Carlo Tree search as the basic technique. It incorporates an explicit stochastic traffic model to use it for load predictions and to prepare the network for upcoming load changes (Figure 54). Basically, FutureCoord takes random samples from the stochastic traffic model, tries to optimize service orchestration along these samples, and picks a most promising action. As expected, the explicit inclusion of these traffic forecasts improves orchestration quality.

Distributed Machine Learning To make these ML approaches usable in a real system, we need to consider where and how to train these models. Transferring all data to a far-away cloud for training to later on retrieve the models is often not practical, given the amount of training data to transport and the frequency of training. Hence, we need to consider techniques for distributed, in-network machine learning. Notably, this problem is a problem in the context of networking for ML (or generally for computing). Here, the questions occurs what networking conditions need to be satisfied such that ML problems can be solved in distributed manner using a parallel computing infrastructure. In contrast, the distributed ML approaches discussed in the previous subsection considered the use of ML for networking, specifically network orchestration.

A key question in such distributed ML setups is whether and how fast a training algorithm converges. To accelerate the algorithm convergence, the space of learning variables is divided into several coordinates and multiple machines are assigned to work on each one asynchronously. The advantage of this approach is the potentially significantly enhanced convergence speed [ZZY+13]. However, the heterogeneity of computing resources and the

assignment of multiple machines to one coordinate induces that each machine effectively computes updates based on information with potentially significant age of information (AoI). The AoI arises because as one machine computes an update, all other machines may update their associated coordinate variables multiple times. It is therefore pertinent to address two problems: 1) When do modern machine learning algorithms work in the presence of the aforementioned AoI? 2) What is a representative model for AoI in asynchronous parallel computing architectures? In [RRK22b] we addressed the first problem. We developed AoI conditions for distributed stochastic gradient descent (DSGD), the main algorithm that underlies deep learning and artificial intelligence. Our conditions relate the cumulative distribution function of the AoI with the learning rates used by DSGD. The relationship shows that for highly parallelized architectures with many asynchronous machines (thus inducing large AoI), the GD updates should be performed with smaller learning rates to counter the error induced by the AoI. In [RRK22a] we addressed the second problem. We proposed a general model for AoI processes using event processes that possess dependency decay. For computing, our AoI model allows modeling of highly correlated traffic that share the parallel machines working on a machine learning problem. In summary, our two works therefore guide the choice of learning rates for machine learning algorithms running on parallel computing systems depending on the degree of correlation of arriving jobs.

Dealing with States   When orchestrating services, an important distinction is whether the components are *stateless* or *stateful*: A stateless component obtains all required information to process a request from the request itself; a stateful component has to remember information from previous requests to process an actual one. Orchestrating stateless components is much simpler as there is no need to keep track of which request flow is mapped to which component, and rerouting can be done arbitrarily. But in reality, stateful components do appear.

One particular challenge is then to ensure that, when a flow is rerouted towards other components, the corresponding flow state is moved along. More specifically, we need to ensure proper timing. Once a request arrives at a new component, the flow state must have already been moved, but it must not be moved before the last request at the old component has not finished processing. It becomes necessary to synchronize flow and state migration with each other.

We tackled this problem by developing SHarP, a seamless handover protocol to integrate flow/state-migration protocol [PKK18b; PKK19] on top of an SDN-enabled network. The key idea is to use the SDN controller as a natural point of serialization by sending a handover message, a very limited number of request messages, and state handling messages via the controller (Figure 55 shows an intermediate step). This does impose additional load on the controller, but in experiments we were able to show that this overhead can be limited to a small number of messages, which should not create an unacceptable performance burden for SDN controllers.

2.2.3 Evaluation & Prototyping

A lot of the orchestration ideas described in the previous subsections were evaluated using simulations that use fairly simplistic models of the underlying system behavior—for
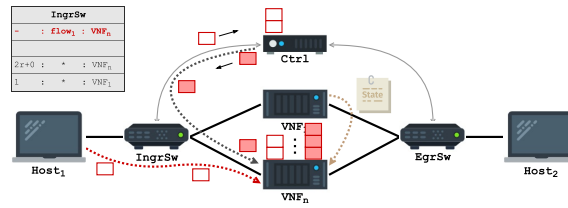
Figure 55: *An intermediate step in SHarP's state handover via ingress switch and SDN controller (Figure 3b of [PKK19]).*

example, the assumption that components of different services do not interact in their resource consumption and that components of the same service have natural dependencies, e.g., that the data rate sustainable by the slowest component determines the bottleneck data rate of an entire service. We were interested in double checking these assumption using actual experiments.

The challenge for such experiments is the required scale: For wide-area and data-center networks, we would need to run experiments on hundreds or thousands of nodes, which might only be feasible in rare circumstances and not amenable to continuous experimental work. Hence, we went for a compromise: to emulate actual environments, but run real code. To do so, we had to extend existing emulation tools. Starting from the well-known MiniNet tool, we first extended it to MaxiNet, enabling it to run in a distributed manner, scaling to thousands of emulated nodes. Then, we added the capability to run ordinary Docker containers as part of that emulation system, published as the tool ContainerNet [PKK18a] (with over 160 forks on GitHub as of late October 2022).

We used ContainerNet to construct a profiling platform for service components and entire services. It did turn out that it is necessary to profile services in their entirety [PK17] to properly reflect their internal interactions. To deal with all these aspects, a non-trivial system architecture emerged (Figure 56).
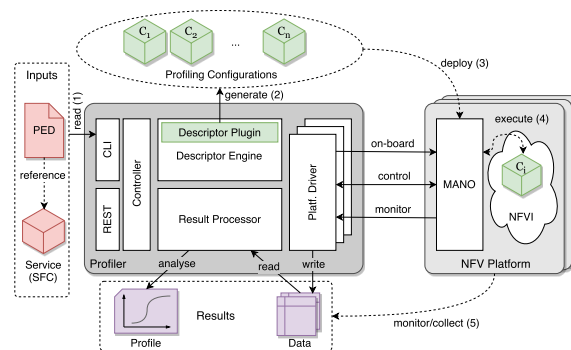
Figure 56: *System architecture of our profiling system interacting with several NFV platforms. The figure also shows the general workflow and generated artifacts (Figure 1 of [PK17]).*

2.2.4 Data Centers

In addition to service provisioning in wide-area networks, we also looked at data center scenarios. For example, we considered how to deal with so-called "coflows": a group

a flow that needs to be complete jointly before a distributed computation can continue its next stage (e.g., in a gather-collect context). We investigated machine-learning-based admission control and resource allocation schemes for that problem.

Here, we would like to describe an older contribution that addresses the following question: How can one generate traffic (e.g., for a simulation or emulation) that faithfully represents key statistical properties of actual data center traces? This is necessary as only limited amounts of traces are available, which is insufficient to drive performance evaluation work that is statistically meaningful.

In Reference [WK16], we describe a traffic generator that serves these needs; its main workflow is shown in Figure 57. Practically speaking, at the time of that work, traces on layer 2 (L2) were available, but for the scheduling work we were interested in, we needed layer 4 (L4) flow traces that were extensible yet faithful. To this end, we analyzed the available L2 traces and tried to infer L4 information from them. Checking whether this inference was correct is simple: Just use these L4 traces to run network experiments, collect L2 information, and compare statistical properties. The question is how to extract L4 information from L2 information, given that L2 information hides the bidirectional nature of TCP (packets are mirrored by acknowledgements in the opposite direction) and that this ACK traffic must not be mistaken for "actual" L4 traffic. We hence had to figure out the distribution functions for packet and ACK packet sizes and to "de-convolute" these different traces from the available L2 information. In the end, it turned out that we were able to construct corresponding L4 traces.
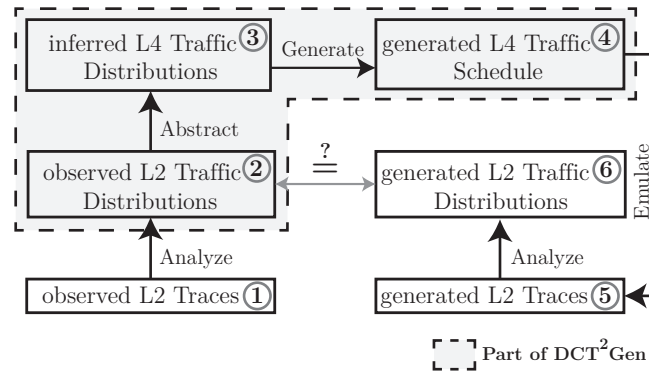
Figure 57: *Workflow of DCT2Gen (Figure 1 in [WK16]).*

3   Concluding Remarks

In Subproject C4, we considered the task of efficiently utilizing resources in highly configurable compute centers, be they big or small, centralized or distributed, from a variety of angles ranging from abstract algorithmic models to concrete framework-specific aspects. We conclude the discussion of these efforts by highlighting possible future research directions for some of the studied topics.

Scheduling with Setup Times

Setup times are extensively studied in the area of scheduling with a wide variety of different models. Regarding the class-based model considered in this subproject [MMMR15; JMM19] there are interesting open problems regarding closely related variants. For instance, the problem of identical machines with preemptions has been studied, i.e., where the processing of jobs may be interrupted and resumed at a later time. However, there is no PTAS known for this setting, and it seems challenging to design one. This is somewhat surprising given the fact that the scheduling problem without preemptions admits a PTAS, and the one with preemptions but without setup times is not even NP-hard. Moreover, considering the variant with preemptions for more general machine models would be interesting as well. Another interesting research direction can be derived from the fact that several novel PTAS results for scheduling with setup times have been obtained via newly developed techniques in the area of integer programming [JKMR22]. These techniques are based on utilizing some structure in the constraint matrix in order to derive provably efficient algorithms. It seems promising to further study the use of these techniques in the area of scheduling and to extend the techniques themselves to enable better or more general results, for instance, regarding scheduling with setup times. Lastly, there has been a recent trend to consider semi-online models in which crucial information regarding the instance is not known in advance, but estimates are given using a machine learning model, for instance. There have been several recent, intriguing results in this direction for scheduling problems presented at high-level conferences. It seems well worth considering scheduling with setup times—or other problems considered in this subproject—from this angle.

Cloud-Assisted Scheduling

As the Internet transforms into a landscape largely dominated by giant cloud service providers, cloud-assisted scheduling has become increasingly important. Since about 2010, there has been a plethora of different models, both practical and theoretical, that try to address some of the challenges that arise from this way of computing. A fundamental problem for theoretical analysis seems to be that there are so many different important properties of scheduling on clouds that a unifying model is currently out of reach. In no particular order, one might consider the leasing model and associated costs, job structure, precedence constraints, communication delays, release times, different machine speeds and capabilities, additional resources, online vs. offline algorithms, and more. Following on from this, it may be interesting to explore the limits at which generalizations of our model from [MMP21] no longer yield efficient (approximation) algorithms. In particular, the rental model and the cost function in our model are rather simple, we can get machines for exactly the time intervals we need, and we pay only in direct proportion to the jobs outsourced. A more elaborate and realistic leasing system for cloud resources, including other costs and start-up times for new machines, could provide interesting insights. Finally, in the context of the CRC, we would like to mention that this issue can also be explored from a market perspective itself, where multiple cloud providers compete to schedule customers' jobs efficiently in order to maximize their own profits.

In-Network Computing

Since the start of this CRC, the notion of in-network computing has changed substantially. By now, it is fairly commonplace to find discussions about many different forms of infrastructure, spreading the resources of data centers ever more thinly across real environments. A common buzzword in this context is the "edge-cloud continuum," where along the path from a device to a centralized cloud, many different forms of service execution opportunities exist, e.g., gateways or micro-cloud data centers of various forms. Somethings, about a dozen or so different stages are differentiated. Often, it is not entirely clear what the differences are, but architectures exist that go to great pains to make such differences and assign different roles, APIs, etc. We believe that artificially introducing differentiation where none exists is detrimental to both the efficiency and uptake of such concepts. We argue that consistent, simple concepts to distribute composed services are to be much preferred, but they have not really materialized, despite a lot of practical progress in using resources of different cloud providers. There is no consistent approach in sight.

We do acknowledge, however, that there are differences in business models associated with such a multi-stage infrastructure. While that certainly drives competition and can be a strong hindrance to standardization towards common APIs, there are also actual consequences. For example, there is no clear notion of a "chain of custody" for storing data or executing services. While fundamentally, this is in many forms unsolvable (essentially, the impossibility of consensus in faulty, asynchronous systems), there still is a need for practical compromises with a clear assignment of responsibilities and custody for data or services. Again, while there is a lot of understanding available about basic concepts and their limitations, there is no agreed standard that would foster the adoption of such architectures.

Prototyping and Actual Experiences

Very much in the same vain, we believe that there is a need for more experimental experience in real systems. A lot of results come from simulation or carefully controlled lab environments and emulations, but there is not much academic work done in real environments "in the wild." As of today, this is still the purview of cloud providers, hyperscalers and major over-the-top providers such as Netflix. This is a particular problem for most work that follows machine-learning approaches: When no data is available, there is nothing from which a model can be learned, and there is even less opportunity to really test approaches to manage data centers (at whatever stage of a continuum). We believe that this methodical gap needs to be closed, but there is no obvious approach how to do that. This is a challenging area for systems research the coming years, which ensures that our work stays relevant by being able to work from and test in relevant environments using relevant data.

Bibliography

[ABC⁺17]   Antoniadis, A.; Barcelo, N.; Consuegra, M. E.; Kling, P.; Nugent, M.; Pruhs, K.; Scquizzato, M.: Efficient Computation of Optimal Energy and Fractional Weighted Flow Trade-Off Schedules. In: vol. 79. 2. 2017, pp. 568–597.

[BBS+00]   BROOKS, D. M.; BOSE, P.; SCHUSTER, S.; JACOBSON, H. M.; KUDVA, P.; BUYUKTOSUNOGLU, A.; WELLMAN, J.; ZYUBAN, V. V.; GUPTA, M.; COOK, P. W.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors. In: *IEEE Micro* 20 (2000), no. 6, pp. 26–44.

[CLL11]   CHAN, H.; LAM, T. W.; LI, R.: Tradeoff between energy and throughput for online deadline scheduling. In: *Sustain. Comput. Informatics Syst.* 1 (2011), no. 3, pp. 189–195.

[DKM18]   DRÄXLER, S.; KARL, H.; MANN, Z. A.: JASPER: Joint Optimization of Scaling, Placement, and Routing of Virtual Network Services. In: *IEEE Transactions on Network and Service Management* (2018)

[GG75]   GAREY, M. R.; GRAHAM, R. L.: Bounds for Multiprocessor Scheduling with Resource Constraints. In: *SIAM J. Comput.* 4 (1975), no. 2, pp. 187–200.

[Gra66]   GRAHAM, R. L.: Bounds for certain multiprocessing anomalies. In: *Bell system technical journal* 45 (1966), no. 9, pp. 1563–1581

[JKMR22]   JANSEN, K.; KLEIN, K.; MAACK, M.; RAU, M.: Empowering the configuration-IP: new PTAS results for scheduling with setup times. In: *Math. Program.* 195 (2022), no. 1, pp. 367–401.

[JL16]   JANSEN, K.; LAND, F.: Non-preemptive Scheduling with Setup Times: A PTAS. In: *Euro-Par 2016: Parallel Processing - 22nd International Conference on Parallel and Distributed Computing, Grenoble, France, August 24-26, 2016, Proceedings*. Ed. by DUTOT, P.; TRYSTRAM, D. Vol. 9833. Lecture Notes in Computer Science. Springer, 2016, pp. 159–170.

[JMM19]   JANSEN, K.; MAACK, M.; MÄCKER, A.: Scheduling on (Un-)Related Machines with Setup Times. In: *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20-24, 2019*. IEEE, 2019, pp. 145–154.

[KK17]   KELLER, M.; KARL, H.: Response-Time-Optimised Service Deployment: MILP Formulations of Piece-wise Linear Functions Approximating Non-linear Bivariate Mixed-integer Functions. In: *IEEE Transactions on Network and Service Management* (2017), no. 1, pp. 121–135

[KMRS17]   KLING, P.; MÄCKER, A.; RIECHERS, S.; SKOPALIK, A.: Sharing is Caring: Multiprocessor Scheduling with a Sharable Resource. In: *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*. Ed. by SCHEIDELER, C.; HAJIAGHAYI, M. T. ACM, 2017, pp. 123–132.

[KP13]   KLING, P.; PIETRZYK, P.: Profitable scheduling on multiple speed-scalable processors. In: *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*. Ed. by BLELLOCH, G. E.; VÖCKING, B. ACM, 2013, pp. 251–260.

[MMMR15]   MÄCKER, A.; MALATYALI, M.; MEYER AUF DER HEIDE, F.; RIECHERS, S.: Non-preemptive Scheduling on Machines with Setup Times. In: *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*. Ed. by DEHNE, F.; SACK, J.; STEGE, U. Vol. 9214. Lecture Notes in Computer Science. Springer, 2015, pp. 542–553.

[MMMR18]   MÄCKER, A.; MALATYALI, M.; MEYER AUF DER HEIDE, F.; RIECHERS, S.: Cost-efficient scheduling on machines from the cloud. In: *J. Comb. Optim.* 36 (2018), no. 4, pp. 1168–1194.

[MMP21]   MAACK, M.; MEYER AUF DER HEIDE, F.; PUKROP, S.: "Server Cloud Scheduling." In: *Approximation and Online Algorithms - 19th International Workshop, WAOA 2021, Lisbon, Portugal, September 6-10, 2021, Revised Selected Papers*. Ed. by KÖNEMANN, J.; PEIS, B. Vol. 12982. Lecture Notes in Computer Science. Springer, 2021, pp. 144–164.

[Pin16]   PINEDO, M. L.: *Scheduling: Theory, Algorithms, and Systems*. Springer, 2016

[PK17]   PEUSTER, M.; KARL, H.: Profile Your Chains, Not Functions. Automated Network Service Profiling in DevOps Environments. In: *IEEE Conference on Network Function Virtualisation and Software Defined Networks (NFV-SDN)*. Berlin, 2017

[PKK18a]   PEUSTER, M.; KAMPMEYER, J.; KARL, H.: Containernet 2.0: A Rapid Prototyping Platform for Hybrid Service Function Chains. In: *4th IEEE International Conference on Network Softwarization (NetSoft 2018)*. Montreal, 2018

[PKK18b]   PEUSTER, M.; KÜTTNER, H.; KARL, H.: Let the state follow its flows: An SDN-based flow handover protocol to support state migration. In: *4th IEEE International Conference on Network Softwarization (NetSoft 2018)*. Montreal, 2018

[PKK19]   PEUSTER, M.; KÜTTNER, H.; KARL, H.: A flow handover protocol to support state migration in softwarized networks. In: *International Journal of Network Management* (2019)

[RRK22a]   REDDER, A.; RAMASWAMY, A.; KARL, H.: Age of Information Process under Strongly Mixing Communication – Moment Bound, Mixing Rate and Strong Law. In: *Proceedings of the 58th Allerton Conference on Communication, Control, and Computing*. 2022

[RRK22b]   REDDER, A.; RAMASWAMY, A.; KARL, H.: Practical Network Conditions for the Convergence of Distributed Optimization. In: *IFAC-PapersOnLine* 55 (2022), no. 13, pp. 133–138

[SJK21]   SCHNEIDER, S. B.; JÜRGENS, M.; KARL, H.: Divide and Conquer: Hierarchical Network and Service Coordination. In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Bordeaux, France: IFIP/IEEE, 2021

[SKK20]   SCHNEIDER, S. B.; KLENNER, L. D.; KARL, H.: Every Node for Itself: Fully Distributed Service Coordination. In: *IEEE International Conference on Network and Service Management (CNSM)*. IEEE, 2020

[SKM+21]   SCHNEIDER, S. B.; KHALILI, R.; MANZOOR, A.; QARAWLUS, H.; SCHELLENBERG, R.; KARL, H.; HECKER, A.: Self-Learning Multi-Objective Service Coordination Using Deep Reinforcement Learning. In: *Transactions on Network and Service Management* (2021)

[SQK21]   SCHNEIDER, S. B.; QARAWLUS, H.; KARL, H.: Distributed Online Service Coordination Using Deep Reinforcement Learning. In: *IEEE International Conference on Distributed Computing Systems (ICDCS)*. Washington, DC, USA: IEEE, 2021

[SSKW19]   SCHNEIDER, S. B.; SHARMA, A.; KARL, H.; WEHRHEIM, H.: Specifying and Analyzing Virtual Network Services Using Queuing Petri Nets. In: *2019 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Washington, DC, USA: IFIP, 2019, pp. 116–124

[WK16]   WETTE, P.; KARL, H.: DCT²Gen: A traffic generator for data centers. In: *Computer Communications* (2016), pp. 45–58

[WSK22]   WERNER, S.; SCHNEIDER, S. B.; KARL, H.: Use What You Know: Network and Service Coordination Beyond Certainty. In: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. Budapest: IEEE, 2022

[YDS95]   YAO, F. F.; DEMERS, A. J.; SHENKER, S.: A Scheduling Model for Reduced CPU Energy. In: *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*. IEEE Computer Society, 1995, pp. 374–382.

[ZZY+13]   ZHANG, S.; ZHANG, C.; YOU, Z.; ZHENG, R.; XU, B.: Asynchronous stochastic gradient descent for DNN training. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 6660–6663

Subproject C5:
Architectural Management of OTF Computing Markets

Gregor Engels[1], Sebastian Gottschalk[1], Dennis Kundisch[2], Christian Vorbohle[2], Nancy V. Wünderlich[3]

1 Department of Computer Science, Paderborn University, Paderborn, Germany
2 Department of Information Systems, Paderborn University, Paderborn, Germany
3 Faculty of Economics and Management, TU Berlin, Berlin, Germany

1 Introduction

Enterprise architectures are used as conceptual blueprints to describe the structure and management of IT systems in organizations. An OTF computer market contains an enterprise architecture that must be continuously aligned with its environment. In times of ever faster-changing environments, architectures must also adapt ever faster with close integration of the business, software, and infrastructure architecture. This section focuses on three aspects of architectural management of OTF computing markets. First, we provide an architectural framework for the static structuring of those markets, including all architecture layers. Second, we focus on the business layer and, in particular, the dynamic business model behavior of the market participants. Third, we identify drivers and barriers to accepting OTF computing markets from multi-stakeholder perspectives. For all aspects, we define the overall research opportunities, show selected highlights of our research, and apply them to the design of OTF computing markets.

The main objective of our research was to understand the design of OTF computing markets from a technical and business perspective. By looking at recent research contributions in the field of enterprise architectures, we recognized that the various research disciplines are more and more intertwined. Here, new substantial contributions result from the structured combination of concepts from existing disciplines. Therefore, we use empirical and conceptual research methods and recombine existing concepts from computer science, information systems, and business administration to develop holistic solutions for OTF computing markets.

As no OTF computing markets currently exist, we conducted our empirical and conceptual studies in comparative markets and discuss how our results apply to the design of future OTF computing markets. Here, we investigated software ecosystems for the architectural framework and business models of single service providers. Next, we discovered business

gregor.engels@uni-paderborn.de (Gregor Engels), sebastian.gottschalk@uni-paderborn.de (Sebastian Gottschalk), dennis.kundisch@wiwi.uni-paderborn.de (Dennis Kundisch), christian.vorbohle@wiwi.uni-paderborn.de (Christian Vorbohle), wuenderlich@tu-berlin.de (Nancy V. Wünderlich)

ecosystems for the interrelationships of different business models. Last, we identified potential drivers and barriers to stakeholders' acceptance of OTF computing markets. Out of that, we derived our corresponding research opportunities.

- **Architecture framework:** Our first research opportunity was to develop an architectural framework for software ecosystems. We first wanted to get an overview of the features of such ecosystems. Based on that, we identified critical design decisions for different architectural layers. Out of that, we derived the main architectural design patterns for various kinds of software ecosystems.

- **Business model development:** Our second research opportunity was to develop business models for ecosystems. We first discovered and analyzed different modeling languages in general and their applicability to business ecosystems. Second, we analyzed different development methods to create a situation-specific business model development approach. Third, we reviewed various software tools for business model development as the foundation for our own one.

- **Attractiveness factors:** Our third research opportunity was to identify success factors for OTF market providers. Based on a literature review on acceptance drivers and barriers from service requesters, service providers, and market provider perspectives, we conducted an exploratory qualitative interview study with potential market participants and market providers.

2   Main Contributions

2.1   Architectural Framework for Software Ecosystems

Nowadays, concerning the changing needs of organizations, simple software solutions have evolved into large-scale software systems [PFG13]. Designing the architecture of those systems in advance is crucial for the success of organizations. However, the designed architectures behind those systems are becoming more complex. Therefore, various architectural approaches are proposed for developing and maintaining enterprise architectures. Among others, the Zachman Framework and the Open Group Architecture Framework (TOGAF) are well-known architecture frameworks for designing software systems. Here, those frameworks structure various layers of the systems, including the business architecture, the software architecture, and the infrastructure architecture. With this, those frameworks aim to provide a closer alignment of the business aspects and technical aspects.

While those architectural frameworks are developed for all kinds of systems, software ecosystems are a subset with special requirements. Here, software ecosystems are defined by Bosch et al. as "*a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs*" [BB10]. However, there is an identified lack of reference models for designing ecosystems that support ecosystem providers in their decision-making [AS16]. Therefore, within our research, we identified the architectural design features of those ecosystems and modeled the variability of possible architectural decisions. Out of that, we extracted patterns for various types of software ecosystems.

2.1.1 Features of Software Ecosystems

To support the decision-making in those ecosystems, we first needed to clarify what the common features of those ecosystems are. We have done that by reviewing the literature on IT service markets in a systematic way.
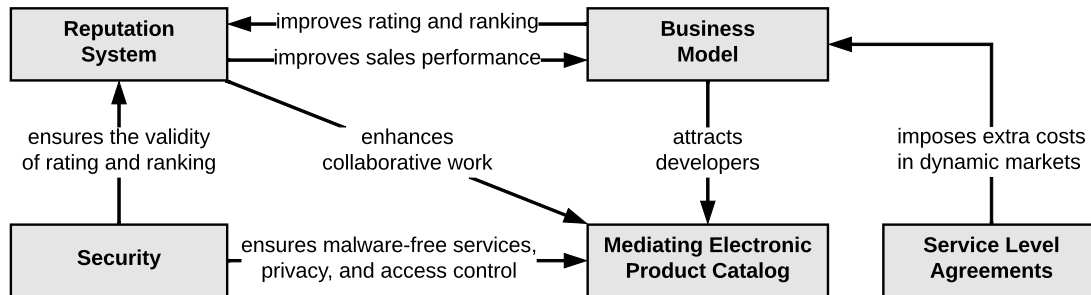

Figure 58: *Primary features of IT service markets [JPEK16].*

The outcomes of our review were the following six primary features, shown in Figure 58, together with several subfeatures [JPEK16]. The *reputation system* is responsible for collecting and aggregating the ratings and reviews of users for the services. Those systems are used to build trust among the different users in the ecosystem and support the ranking of single services. The *business model* is used to describe the rationale of how the ecosystem can create, deliver and capture value for its users. This, in turn, supports the sustainable growth of the ecosystem over time. The *recommendation system* handles the discovery of different services within the ecosystems. With this, the ecosystem ensures the users' acceptance of new services. The *mediating electronic product catalog* acts as an intermediary between the users and the developers to provide the users' access to the services. The ecosystem uses a catalog to support the standardized discovery and delivery of those services. The *security* is used to save the users' privacy and analyze the developers' code. This is important for creating trust among the users for the provided services. The *service level agreements* are used to guarantee the quality of certain services. This ensures the usability of services also in a business context. We created a variability model for designing software ecosystems from those identified sources and initial features.

2.1.2 Design Variabilities for Software Ecosystems

To support the decision-making for new and existing ecosystems, variability models represent alternative design decisions, including various variation points with different variants. To derive such a variability model, we used a systematic taxonomy development method, derived a taxonomy for classifying objects based on their common characteristics, and mapped it to a variability model.

The variability model, shown in Figure 59, consists of three different views based on the architectural layers of enterprise software systems [JZEK17]. For the visualization, we use orthogonal variability models, where mandatory or optional variation points for the layer are connected through a minimum and maximum of mandatory or optional variants as corresponding choices.
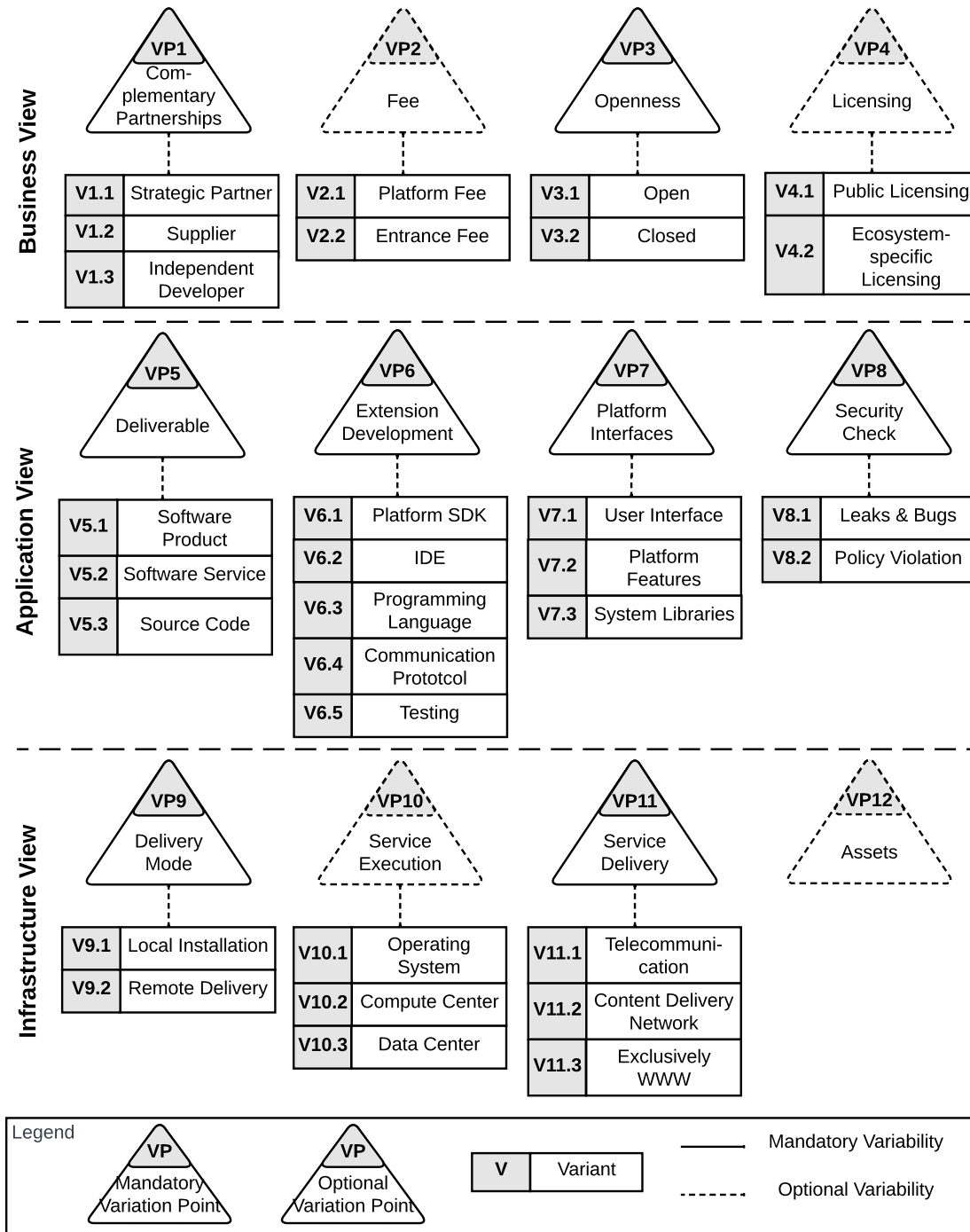
Business View

VP1 Complementary Partnerships	
V1.1	Strategic Partner
V1.2	Supplier
V1.3	Independent Developer

VP2 Fee	
V2.1	Platform Fee
V2.2	Entrance Fee

VP3 Openness	
V3.1	Open
V3.2	Closed

VP4 Licensing	
V4.1	Public Licensing
V4.2	Ecosystem-specific Licensing

Application View

VP5 Deliverable	
V5.1	Software Product
V5.2	Software Service
V5.3	Source Code

VP6 Extension Development	
V6.1	Platform SDK
V6.2	IDE
V6.3	Programming Language
V6.4	Communication Prototcol
V6.5	Testing

VP7 Platform Interfaces	
V7.1	User Interface
V7.2	Platform Features
V7.3	System Libraries

VP8 Security Check	
V8.1	Leaks & Bugs
V8.2	Policy Violation

Infrastructure View

VP9 Delivery Mode	
V9.1	Local Installation
V9.2	Remote Delivery

VP10 Service Execution	
V10.1	Operating System
V10.2	Compute Center
V10.3	Data Center

VP11 Service Delivery	
V11.1	Telecommunication
V11.2	Content Delivery Network
V11.3	Exclusively WWW

VP12 Assets

Legend:
- VP Mandatory Variation Point
- VP Optional Variation Point
- V Variant
- ——— Mandatory Variability
- - - - - Optional Variability

Figure 59: *Variability model of architectural design decision [JZEK17].*

The **Business View** includes the most influential decisions of the business strategy to create a value-capturing environment around the ecosystem. Here, the *complementary partnerships* defines a strategy to choose partners for adding additional services to the ecosystems. *Fees* describe the provision of payments to enter the ecosystem and use corresponding services. *Openness* defines a strategy to which degree access to the ecosystem is possible for the participants. *Licensing* describes how the ecosystem and additional services are licensed to the partners as well as to the participants of the ecosystem.

The **Application View** includes the architectural decisions focusing on the extensibility of the ecosystem by the complementary partners. Here, the *deliverable* provides different types of artifacts for the services the participants could use. *Extension development* provides various techniques to allow partners to develop and test services for the ecosystem. *Platform interfaces* provide various gateways to integrate the developed services into the ecosystem. *Security checks* are integrated into the development or the delivery to protect the participants of the ecosystems from misuse.

The **Infrastructure View** includes the necessary hard- and software to realize the functionalities of the application view. Here, the *deliverable mode* provides different options for delivering the services to the participants. The *service execution* describes the location where those services are actually executed. The *service delivery* shows the backend technology that the ecosystem provider uses to run the ecosystem and deliver the services. The *assets* are additional devices that are used by the ecosystem provider to deliver the ecosystem to the participants. Based on those variabilities, we extract architectural patterns of typical software ecosystems.

2.1.3 Architectural Patterns for Software Ecosystems

To support the decision-making for those ecosystems, patterns describe abstracted knowledge that occurs in multiple organizations. We derive those patterns using a quantitative pattern-extraction method.
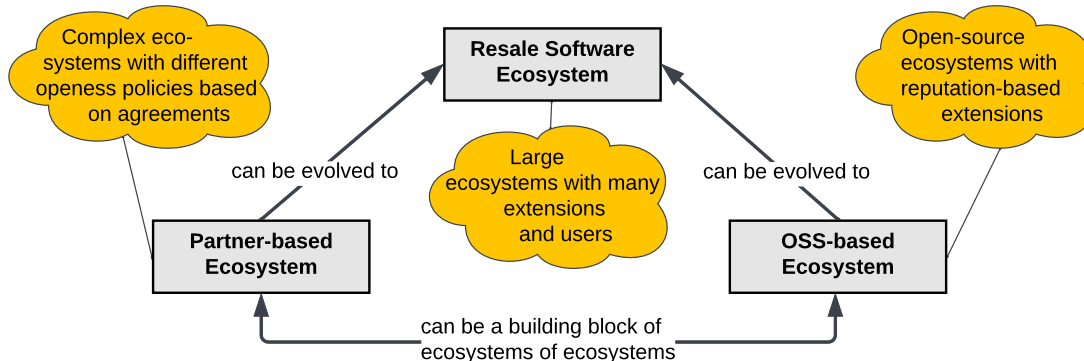
Figure 60: *Architectural patterns of software ecosystems [JZK+18].*

As shown in Figure 60, for the outcome we identified three different patterns of software ecosystems [JZK+18]. Here, *resale ecosystems* provide a large number of extensions by different independent external developers. After their creation, the extensions are sold to many users within the ecosystem. Next, *partner-based ecosystems* are used for complex ecosystems in new industrial sectors, where the external developers and ecosystem providers build new extensions based on partnership agreements. Here, different openness policies support providers in protecting intellectual property within their ecosystems. Last, in *OSS-based ecosystems*, the software platform is mostly released under open source by the ecosystem provider. The external developers are primarily not financially motivated to develop extensions. Instead, they aim to gain reputations or extend the ecosystem for

their own purposes. Over that time, the partner-based and OSS-based ecosystems might evolve into resale ecosystems. By considering those results, we developed our architectural framework for software ecosystems.

2.1.4 Application to OTF Computing Markets

Architectural design decisions are essential for ecosystem providers in order to create new or revisit existing ecosystems. However, there was less structured information on the most critical design decision for the different architectural layers of business, software, and infrastructure of software ecosystems available. Therefore, we developed an architectural framework consisting of ecosystem market features, architectural design variabilities for all layers, and possible ecosystem patterns.

Out of that knowledge, we developed an open-source software tool called SecoArc [SE20] as an Eclipse plugin, which contains two main components for the pattern-centric design of software ecosystems. First, ecosystem providers can model the different variabilities of their ecosystems under the consideration of an existing meta-model. Second, the provider can analyze those decisions for conformance errors and receives suggestions for architectural patterns. With the tool, we support ecosystem providers in creating and improving their software ecosystems. Here, we applied our architectural framework to design OTF computing markets. By using the definition of software ecosystems from Bosch et al. [BB10], the OTF market provider might provide the software platform for the ecosystem. In those ecosystems, OTF service providers are the external developers whose services are composed by the OTF providers as domain experts to individual solutions. Those composed services are used by the OTF requestors as users. Lastly, the services are executed in the infrastructure of the OTF compute center. Therefore, we can apply our architecture framework for the market provider to design those ecosystems.

We applied our software tool SecoArc to the OTF Proof of Concept (OTF-PoC). Here, the OTF-PoC[21] is an instance of a potential OTF computing market, where a chatbot interface is used to configure AI-based services. We conducted a case study with the aim of opening the ecosystem for external services by placing representatives of the PoC development team as well as external service providers in the role of potential market providers. The market providers use the SecoArc tool to model different variabilities of the ecosystems. Those variabilities consist of business-related (e.g., free or paid entrance fee (V2.1 in Figure 59)), application-related (e.g., Java and Python as programming languages (V6.3)), and infrastructure-related (e.g., single or multiple servers for remote delivery (V9.2)) design decisions.

Out of those designed decisions, they derived two different architectures. Here the overall vision was to provide "*an ecosystem [that] should support innovation while being sustainable in terms of confronting external threats that could have a long-term impact on the platform's success. Furthermore, the platform ownership should be managed by using the GNU General Public License (GPL)*." The first one is an open ecosystem in which the platform remains an open-source project so that the ecosystem grows through

---

[21]Website of the OTF-PoC: `https://sfb901.uni-paderborn.de/projects/tools-and-demonstration-systems/tools-from-the-2nd-funding-period/proof-of-concept`

the direct contributions of service providers and the source code can be freely used. This mostly relates to the OSS-based ecosystem architecture design pattern. The second one is a semi-open controlled ecosystem where the number of service providers on the platform drastically increases and the openness needs to be reduced by controlled software development and the marketing environment. This mostly relates to the resale ecosystem architecture design pattern. Based on those architectural patterns, we also need to develop corresponding business models for actors in the business ecosystem. While we were working on the development of the framework and its application to OTF computing markets, we saw a special need for investigating the business models as part of the business architecture.

2.2 Business Model Development for Ecosystems

Business Model Research is a rapidly growing field, and the concept's usefulness has been emphasized in research and practice. For the case of OTF computing, the interplay of business models and technology is especially crucial as "*a mediocre technology pursued within a great business model may be more valuable than a great technology exploited via a mediocre business model*" [Tee10]. Consequently, the concept of business models is a well-established means to offer an essential contribution to the business architecture and acts as an intermediary between business strategy and business processes. A business model describes the design or architecture by which a company creates and delivers value for its customers, thereby generating profit [Tee10]. Some business model definitions also make references to representing a system. By that, business models can also be viewed as a system consisting of activities performed by the company and its ecosystem partners [ZA13]. The challenge in developing business models for IT ecosystems is that every combined product or service can be commercialized via an uncountable number of possible business model alternatives. Developing economically successful business models for all participants in a complex system, such as OTF computing markets, is a substantial challenge and a decisive success factor.

Ecosystem Research is currently receiving increased attention from research and practice. The concept of ecosystems originated in biology and was adapted to the business context in the early 1990s as a community of cooperating companies or individuals. This community creates products and services for customers who are also part of the business ecosystem, such as suppliers, competitors, and other stakeholders. More recent research defines a business ecosystem as companies collaborating to create a focal value proposition [Adn17], consisting of multilateral and non-generic complementarities coupled with the absence of full hierarchical control [JCG18]. This means that what enables businesses to collaborate and align in an ecosystem is creating a shared value proposition for customers. The business ecosystem concept, therefore, calls for a multi-actor assessment of how value is created, delivered, and captured, i.e., an evaluation of whether a viable business model is established for each ecosystem participant.

Both concepts provide the foundation for our second research opportunity: the development of business models for IT ecosystems. However, akin to technology innovation, creating and innovating a business model within an ecosystem environment is a creative and collaborative process. Therefore, starting from the business model and business ecosystem

concept, we first analyzed and further developed modeling languages, innovation methods and software tools to overcome knowledge boundaries, depart from traditional business approaches, and utilize innovative ways instead to create, deliver and capture value.

2.2.1 Languages for Business Model Development

Business model modeling languages (BMMLs) explicitly communicate the core logic and elements of a business model and employ "*semantic constructs, visual form, and visual notation to represent the business model of a given organization (but not tied to any specific organization) for one or more purposes and through a consistent set of rules*" [SMJ+22].

Compared to other subfields of research on conceptional modeling (e.g., process modeling), research on business modeling is a relatively young subfield of research and less studied. Given the continuously increasing relevance of business model innovation, we argue that BMMLs will continue to be a relevant research topic for different research disciplines such as computer science, information systems, or strategy. However, research within and across disciplines has remained disparate rather than cumulative. Limited knowledge accumulation is problematic because single contributions tend to remain isolated with little relation to other solutions. The current proliferation of BMMLs substantially aggravates the development of a cumulative research tradition. This underlines the necessity for a firm understanding of the state of the art of modeling languages for business models and a respective research agenda. The primary aim of this identified research opportunity is (I) to advance our understanding of business modeling in general and, more specifically, (II) to determine how companies in business ecosystems can be supported in developing innovative business models collaboratively.

(I) Advance general understanding of business modeling. We observed that general knowledge of BMMLs was limited in at least five major ways: (1) limited consolidation (What BMMLs exist?); (2) limited theoretical grounding (How can we compare different BMMLs?); (3) limited evaluation (What are their similarities and differences?), (4) limited understanding of use (How were they researched and used so far?) and (5) future research opportunities (What do we still need to know?). To answer these questions, we conducted a cross-disciplinary synthesis of widely used BMMLs [SMJ+22].

Limited consolidation. In total, we identified 17 different BMMLs. The most well-known examples are the Business Model Canvas [OP10] and e3value [GA03]. In Figure 61, we demonstrate both modeling languages with an example of a mobile app store. Other examples of BMMLs include the Causal Loop Diagram, ebusiness model schematics, the Strategic Business Model Ontology, and the Value Stream Map.

Limited theoretical grounding. We suggest that BMMLs can be analyzed in terms of three main characteristics: i.e., a) content, b) visual notation and form, and c) context of use, also referred to as semantics, syntax, and pragmatics. The semantics of a modeling language refers to what a language attempts to represent (i.e., the "vocabulary"). Syntax refers to how a modeling language represents content, i.e., the type of visual notation it uses (i.e., graphical symbols) and the type of visual form it takes (i.e., the architectural form of a representation). The pragmatics of a language refers to the context of use under which a modeling language is applied.
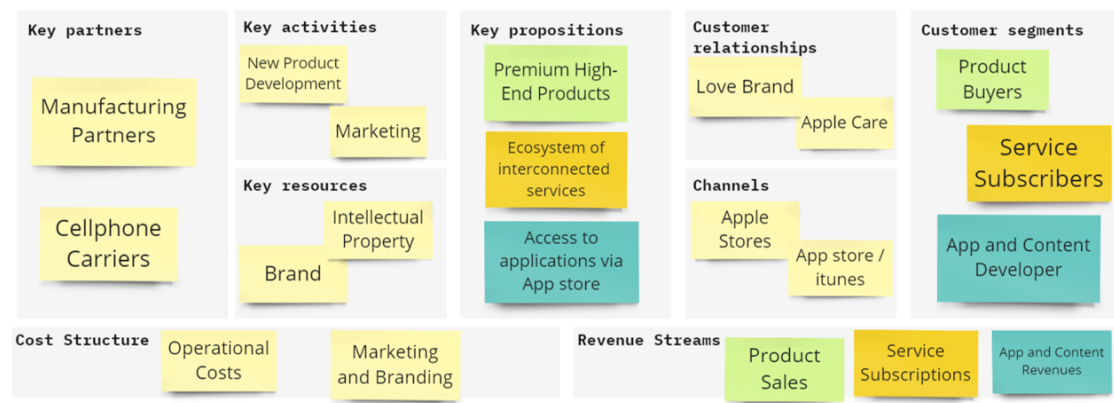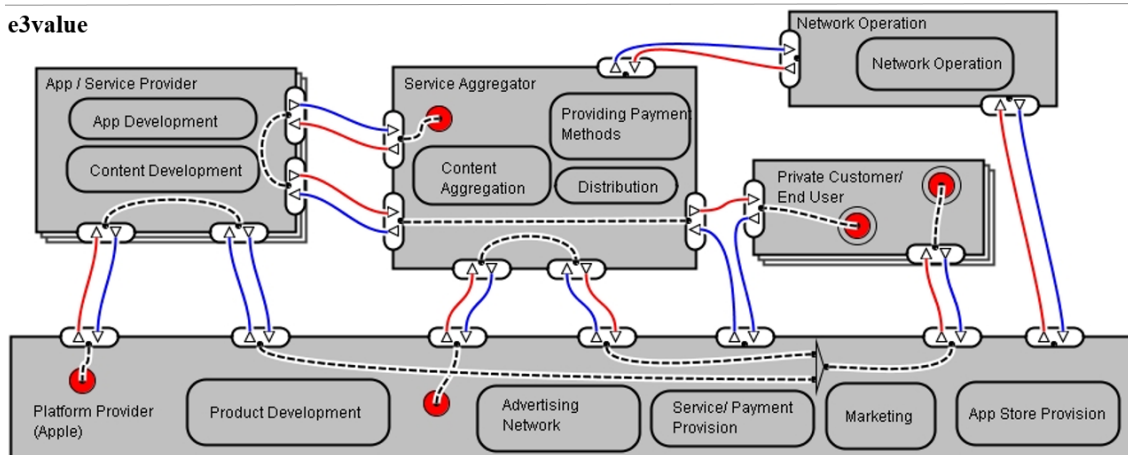
Business Model Canvas

Figure 61: *Two different ways of visualizing a mobile app store business model.*

Limited evaluation. Comparing the identified BMMLs in terms of semantics leads to the identification of different levels (low, moderate, high) of granularity and scope. For example, the Business Model Canvas has a moderate scope and granularity because, with its 9 semantic constructs, it covers 11 semantic sub-dimensions. In contrast, the e3value covers fewer than 9 semantic sub-dimensions and therefore has a lower scope and granularity, just as most other BMMLs. In terms of syntax, the majority use a network-based visualization approach (e.g., e3value), and only three use a map-based approach (e.g. Business Model Canvas). In terms of pragmatics, we identified five main purposes (e.g., generate business model ideas) by analyzing the author's intention to use BMMLs. Here, the intention to use the Business Model Canvas includes all five purposes. In comparison, e3value only includes 4 purposes because it does not intend to support the design of software-based business model development tools. In summary, our identified BMMLs have been developed in a variety of disciplines and for different purposes. No well-accepted set of semantic constructs exists, and various visual notations with a varying number of views for representing semantic constructs have been proposed.

Limited understanding of use. To answer this question, we first analyzed research with BMMLs in greater detail. Summarizing the purposes of employing BMMLs for research, there is no systematic coherence between BMMLs and the purpose for which they are used. However, very different BMMLs are used but for different purposes, partly also within the same research discipline. Nevertheless, we identified four different ways of how the

purposes described before are realized with the help of a BMML: (1) artifact development, (2) data collection, (3) business model analysis, and (4) results communication. Regarding research about BMMLs, we identified multiple studies across four research streams. The majority researched the Business Model Canvas, followed by e3Value. Overall, our review reveals eight purposes for conducting research about BMMLs, e.g., for supporting the development of software or design business models for sustainability. Across the eight identified purposes, there are three different ways of how these purposes are pursued to research about BMMLs: studies that (1) link BMMLs with other modeling languages, (2) extend BMMLs with respect to the respective disciplinary background, (3) theoretically ground BMMLs (e.g., [SL20]).

Future research opportunities. We took advantage of the opportunity that this thorough analysis offers to suggest avenues for moving forward. For that, we used the same dimensions to compare BMMLs, namely semantics, syntax, and pragmatics. Future research should bridge existing knowledge and integrate investigations in areas such as creativity and innovation management, information systems, or marketing and strategy. These areas have the potential to contribute to research with and about BMMLs. Moreover, two emerging research directions offer great potential for further investigations namely digitally enabled business models and sustainability. Figure 62 offers an illustration and synthesis.

| | Research findings What do we know? | | Research gaps What do we need to know? | Opportunities for future research | | |
				Research challenges What challenges need to be overcome?		Research directions
Semantics (Meaning)	Variety of semantics: Partially complementary, partially conflicting	Research *with* and *about* BMMLs	Lack of a well-accepted semantic foundation	Semantics of the business model concept are still debated		Bridging: Integrate/import knowledge from… Creativity and innovation management: - Idea generation experiments - Expert evaluation - Design knowledge for software tools for new product development Information systems: - Modeling language research - Design knowledge for creativity support systems/electronic brainstorming systems Marketing and strategy: - Observational studies/ qualitative field research - Visual analysis Emerging themes Digitally enabled business model and digital offerings Sustainability
				Difficultly to determine the semantic correctness of a business model		
				Difficulty to determine the quality of a business model		
Syntax (Visual form)	Variety of syntax: Partially complementary, partially conflicting		Lack of a well-accepted syntactic foundation	Syntax differently impacts the subjective and the objective usefulness		
Pragmatics (Use context)	Five main purposes: (1) Understand/communicate (2) Analyze/evaluate (3) Deduce requirements (4) Generate ideas (5) Support software tools		Lack of a well-accepted set of context factors	Multiplicity of use contexts	Visualizing versus visualization	

Figure 62: *Summary of research on business model modeling languages [SMJ⁺22].*

(II) Advance a specific understanding of business modeling for business ecosystems.
Based on these findings and identified future research directions, in [VK22] we analyzed the merits and limitations of existing BMMLs to design and analyze business ecosystems. In terms of semantics, we focused on analyzing relationship types to describe different possibilities for connecting business models or business model components (e.g., structural, dependency, dynamic). The multitude of BMMLs appear network-based and represent interactions between actors in a business network. While most BMMLs in this segment are limited to exchanging value streams, others include intangible exchanges such as knowledge, internal impacts and goals, resources, and processes. Moreover, we built on

existing ecosystem theory, for example, from [Adn17] and [JCG18], and derived four dimensions for ecosystem analysis: (1) contingency risk, (2) specificity, (3) governance, and (4) dynamics.

Despite network-based BMMLs being strongly related to enterprise interaction, BMMLs need to pay more attention to the visualization of a shared value proposition and are therefore less suited to analyzing the characteristics of companies collaborating in the same business ecosystem. In general, our analysis shows limited possibilities to analyze business ecosystems from the perspective of ecosystem theory.

By advancing both the general and specific understanding of BMMLs we provide a starting point for further systematic comparisons and build on these findings by creating a modeling approach that sets the focal value proposition in the center of the visualization. In this way, we enable business modeling not only for intra-organizational business model innovation but also for inter-organizational alignment, as in the case of OTF computing markets. With our extensive knowledge base on BMMLs, future research holds great potential to enable collaboration, thus contributing to the next steps of business modeling.

2.2.2 Methods for Business Model Development

To develop business models, domain experts in research and practice have proposed various business model development methods (BMDMs) with different levels of detail in their usage. To support the business developer in using these levels, we have conducted a design science research study to propose a situation-specific business model development approach that composes BMDMs to a specific situation of the organization. This situation-specific adaptation has already proven its value in Situational Method Engineering (SME) [HRÅR14], in which situation-specific software development methods are constructed from fragments of a method repository. Our approach, as shown in Figure 63, introduces five roles that are centered around three stages [GYNE22a].

The first stage of **Knowledge Provision of Methods and Models** is used to utilize the knowledge about the development methods of the business model development and the (canvas) models to visualize of the (business) models. In the beginning, the *meta-method engineer* creates meta-models for the repositories of the methods and (canvas) models *(1.1)*. Here, the *method repository* is able to store different development steps together with development phases or development step sequence patterns for later structuring. Moreover, the *(canvas) model repository* is able to store different models of canvasses and templates together with predefined information on them. Next, different *domain experts* explain their domain knowledge of existing development methods and (canvas) modeling artifacts to the *method engineer (1.2)*. The *method engineer*, in turn, formalizes that knowledge in terms of development methods and within (canvas) models according to the meta-models to make it usable within the approach *(1.3)*.

The second stage of **Composition of Development Methods** is used to construct the *development method* out of the *method repository* and link method steps to the different *(canvas) models* in the *(canvas) model repository* that are used within the development. Here, the *business developer* of the organization explains to the *method engineer* the current context in terms of the situation of the organization and application domain of the service in which the business model should be developed *(2.1)*. The *method engineer*
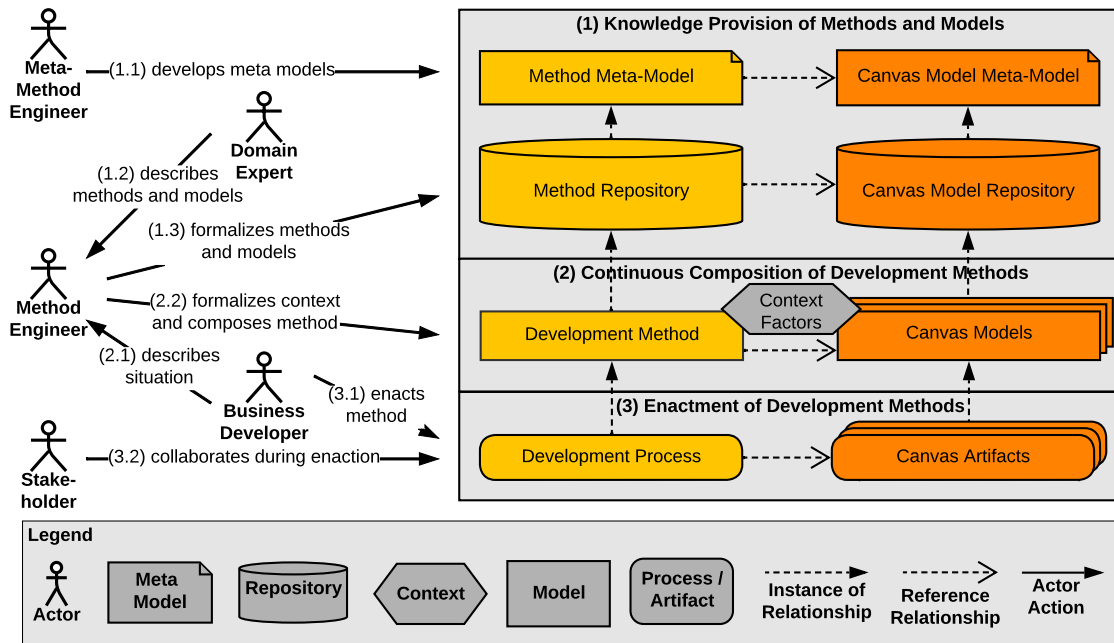
Figure 63: *Situation-specific development of business models [GYNE22a].*

formalizes these *context factors* as the situation of the *development method* and application domain of the *(canvas) models* together with composing the *development method (2.2)*. Here, we support the pattern-based and phase-based composition of development methods. For the pattern-based composition, we select different patterns based on the situation from the *method repository* and nest them into each other. After that, we fill placeholders in those patterns with development steps that are also selected concerning the situation. For the phase-based composition, we select the different phases we want to support from the *method repository*. After that, we select development steps for each phase concerning their situation and order their execution sequence. For both compositions, we connect single development steps to canvas models or template models in the *(canvas) model repository* that are selected concerning the application domain of the service.

The third stage of **Enactment of Development Methods** is used to execute the *development process* and create and modify corresponding *artifacts* during the development. Here, the *business developer* executes the constructed *development method* as a *development process* and uses the linked *(canvas) models* as *(canvas) artifacts* like for the canvasses or templates *(3.1)*. Moreover, other *stakeholders* can contribute to different development steps and modify *(canvas) artifacts* during the execution *(3.2)*. Here, that execution can also lead to a change of the context and, therefore, to a modification of the development method.

We applied our approach in two different domains. In the first domain, we analyzed gray literature to develop business models for mobile applications [GYNE21]. Here, the development methods are structured according to identified method patterns and canvas models are used during the enactment. In the second domain, we analyzed design thinking techniques for conducting design thinking workshops [GYNE22c]. Here, the development methods are structured according to different design thinking phases and predefined whiteboard templates are used during the enactment. To support the situation-specific

development of business models, we have implemented the whole approach in a software tool.

2.2.3 Software Tools for Business Model Development

Various software-based business model development tools (BMDTs) have been developed in research and practice to support the development of business models. To get an overview of the functionalities of those BMDTs and identify open research topics, we used a systematic taxonomy development to derive an underlying taxonomy of BMDTs functionalities.

	Dimension	Characteristics				
Modeling	**Customization**	Add	Divide	Link	Rename	Change arrangement
	Development	Element		Element connection		Template
	Commenting and Linking	Textual comments on element-level	Textual comments on business model-level	Graphical comments (predefined)	Graphical comments (freedom)	Link files · Link web-resources · Glossary support
	Assessment	Financial		Non-financial	Assessment status	Correctness checker
	Navigation and Filtering	Model comparison	Element filter	Phase management	Element clipboard	Link to business model · Framework support
Collaboration	**Communication**	Chat		Discussion board		User list
	Synchronization	Asynchronous modeling		Concurrent modeling		Synchronous modeling
	Assessment	User management	Role management		Support of task sharing	Workspace awareness
	Communication	Version control		Local repository		Remote repository
Technical	**Architecture**	Client / Server		Client only		Web-based
	Data exchange	Export			Import	

Figure 64: *Functionality taxonomy for business model development tools [SSJ+20].*

The first outcome is the **Functionality Taxonomy** with the following three perspectives [SSJ+20] as shown in Figure 64. The *modeling* perspective refers to functions used particularly during the creation of a business model. Those functionalities range from the customization of the underlying models, over commenting and linking on boards, to navigation and the filtering of instances. The *collaboration* perspective presents functions that support collaboration during the business model development. We derive functionalities such as communication through the users, the synchronization of the modeling, or the management of users and roles. The *technical* perspective describes the technical attributes of those tools. It consists of the communication architecture of those BMDTs together with their exchange of data.

The second outcome is a **Future Research Agenda** with the following five topics. This agenda is structured around the groups of future functions, the evaluation of performance, the incorporation of user and task characteristics, and the methods used. Here, the group

of methods topics refers to the variety of ways a BMDT for business model development can be used. Here, its usage can be ranged from micro-level (e.g., the decision for a moderating team member) to macro-level processes (e.g., usage of a specific development method) or specific to the reasons for an organization (e.g., developing a new business model vs. improving an existing business model). One open challenge is the composition and enactment of business model development methods according to the situation of the organization.

Based on those functionalities and open research in the method group, we developed the *Situational Business Model Developer (SBMD)* [GYNE22b] as a BMDT, which was derived as an IT artifact from the DSR study for situation-specific business model development [GYNE22a]. It is a web-based solution that can be used within the web browser,[22] and the source code is freely accessible.[23] It supports all the stages of knowledge provision of methods and models, the composition of development methods, and their enactment. The tool can be used by a single user or multiple users can collaborate during the development steps. Moreover, the tool is based on a modular architecture, making it extensible for new methods, models, and development support techniques.
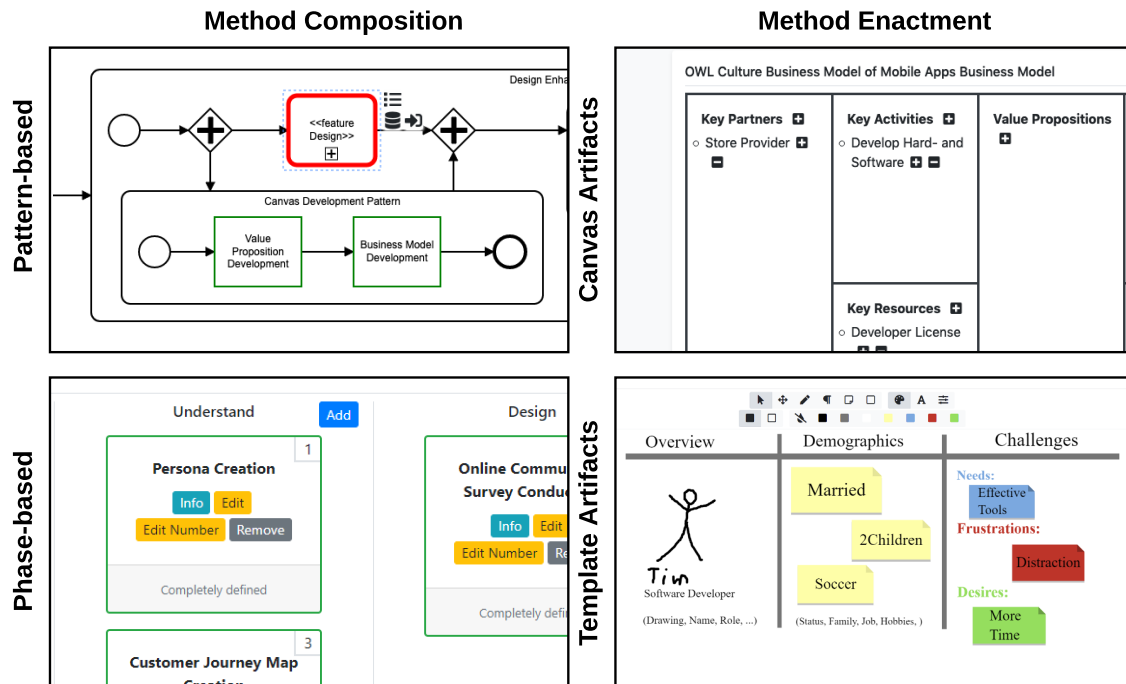

Figure 65: *Screenshots of the Situational Business Model Developer.*

Parts of our **Situational Business Model Developer** for the composition and enactment of development methods can be seen in Figure 65. Before the composition and enactment, our tool provides the knowledge of methods and models by filling the predefined repositories. We filled those repositories with knowledge about business models for mobile apps and workshops for design thinking. Nevertheless, those repositories can be continuously

---

[22]Online version of the Situational Business Model Developer: `http://sebastiangtts.github.io/situational-business-model-developer/`

[23]Source code of the Situational Business Model Developer: `https://github.com/sebastiangtts/situational-business-model-developer`

extended by the users of the tool. For the method repository, we allow the creation of atomic method elements combined into method building blocks as development steps and optionally structured according to method patterns. For the (canvas) model repository, we allow the creation of atomic (canvas) elements that are structured into (canvas) building blocks and visually represented through (canvas) models. For the *method composition*, and after defining the context, we support using patterns and phases. During the *pattern-based* composition, we support the combination of different method patterns to structure the method building blocks. During the *phase-based* composition, we support selecting different phases from the method elements and assign of multiple method building blocks to them. For the *method enactment*, we support the creation of canvas artifacts and template artifacts, besides from simple text documents. The *canvas artifacts* are visual representations of canvas models where predefined knowledge of the canvas building blocks supports filling out of the boxes. The *template artifacts* are visualizations of the template models, which can be freely filled out. During the enactment, the conduction of development steps might also lead to a change in the context and, therefore, a change in the composed development method. A deeper explanation of the tool is available in the explanation section within the tool. By considering those results, we draw the lessons learned for ecosystem business model development.

2.2.4 Application to OTF Computing Markets

The concept of business models and business ecosystems are well-established means to contribute to creating a business architecture. Using these concepts, we analyzed BMMLs, innovation methods, and software tools in general, identified current research gaps, and proposed a research agenda with future research directions. Building on this research on modeling languages, methods, and tools, we developed a procedure model for business model collaboration, such as in the case of OTF computing markets.

In [RLL+22], we offer a comprehensive new set of methods and modeling approaches, which should be used in a workshop setting with multiple company representatives and can be supported by collaboration tools such as Miro or as a part of design thinking workshops in our SBMD. The BMI4BE procedure model can be used for two possible application contexts: (1) to innovate an already existing business ecosystem or (2) to design a new business ecosystem. By this, the procedure model is also an extension of existing business model innovation methods (e.g., The Business Model Canvas).

The idea of the procedure model (see Figure 66) is based on the consideration that representatives of the individual companies first focus on creating value for potential customers of the business ecosystem to develop a shared value proposition and a market perspective. Subsequently, the (potentially) participating companies' contributions to the ecosystem are specified, and the necessary business model flows are developed. On this basis, a common visualization of the ecosystem is created to analyze the flows between the individual companies. In the last step, every company uses these results to develop the necessary transformation for their current business model. These steps can be repeated iteratively until every representative agrees on the outcome.

Figure 67 shows an example of the visualization of a possible ecosystem flow map (Phase 5) for OTF computing markets. This example was created during a CRC901 research
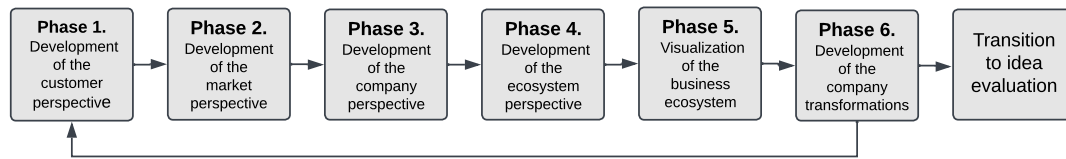
Figure 66: *Phase diagram of the BMI4BE procedure model [RLL+22].*

seminar in May 2022 with researchers from other subprojects and is one of four innovative solutions created during a workshop. The example in Figure 67 shows that OTF computing markets could serve more than one customer segment, which mainly interacts with a market provider. Behind this market provider, OTF computing solutions are generated by an ecosystem of multiple OTF providers, which combine modules from the service provider, infrastructure provider and component provider. Moreover, computing centers deliver fast cloud solutions for small and medium-sized enterprises. As this example shows, the market provider and its attractiveness to potential customers play a central role. These attractiveness factors are further analyzed in the next section.
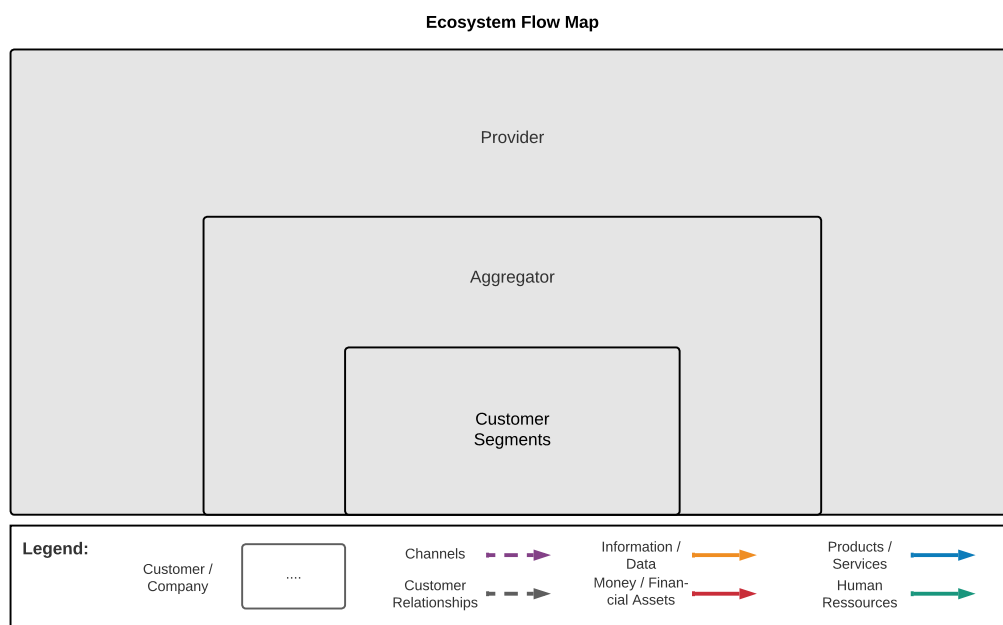

Figure 67: *Ecosystem flow map of an OTF computing market.*

2.3 Attractiveness of Platforms

Platforms can be viewed as particular kinds of markets that play the role of facilitators for an exchange or a transaction between different types of stakeholders that could not otherwise, transact with each other, or only with great difficulty [BKW21]. We define a digital platform as a mediating entity operating in two (or multi)-sided markets, which uses the Internet to enable direct interactions between two or more distinct but interdependent

groups of users so as to generate value for at least one of the groups.

In our research, we explore the success factors of digital platforms based on a literature review on drivers and barriers of technology acceptance in related contexts and on an empirical qualitative research study with potential stakeholders of a future OTF market.

2.3.1 Stakeholders of Matchmaking Platforms

Our research focuses on the application of the OTF market as a type of matchmaking platform such as Airbnb and TaskRabbit, which are often characterized by "*high platform intermediation as well as high levels of consociality*" [PK18]. Matchmaker platforms provide the infrastructure to facilitate interactions among the platform users: the service providers on one side and the service requesters/end users on the other. Market providers of such matchmaking platforms typically do not provide the services themselves that are offered on the platform (e.g., only hosts offer accommodation). The core value proposition of a matchmaking platform provider comes from providing a pairing of market participants (e.g., matching hosts with guests [PK18]). Thus, market providers constitute new organizational forms that rely on individual service providers as their co-producers, who are not employees of the platform, but usually act as independent entrepreneurs. On matchmaking platforms, we observe triadic service relationships between three stakeholder entities: the market provider, the service providers, and the service requesters.

Thus, the relationship management measures of a platform provider differ considerably from the relationship management of traditional customer-firm relationships given the unique characteristics of matchmaking platform business models. A market provider not only has to manage a requester base and attract and retain a critical mass of requesters on the platform but also needs a critical mass of service providers that guarantee the service provision and consumption on the platform. Thus, it is crucial for market providers to understand what drives the attraction of matchmaking platforms in order to recruit and retain participants on all market sides.

2.3.2 Framework of Drivers and Barriers of Platform Acceptance

Literature on platform acceptance is sparse and often reflects only one stakeholder perspective (e.g., end user's perspective). For example, [JPML21] conducted a study with 450 Airbnb guests and identified network externalities, trust, perceived ease of use, perceived usefulness, and the level of interactivity on the platform as factors influencing end users' intention to purchase on the Airbnb platform. Since there is no study that brings together the acceptance drivers and barriers from multiple stakeholders, we develop a framework on multi-stakeholder perspectives on the acceptance of matchmaking platforms. Figure 68 illustrates such a framework.

To develop our framework, we draw from different literature streams that explore potential acceptance factors pertaining to one or more perspectives. In particular, we draw on literature on technology acceptance from an end-user and employee perspective (e.g., [Dav89]), literature on the digital transformation of firms (e.g., [WLCH19]), and literature on work-relationships of employees and solo entrepreneurs (e.g., [KBE21]).
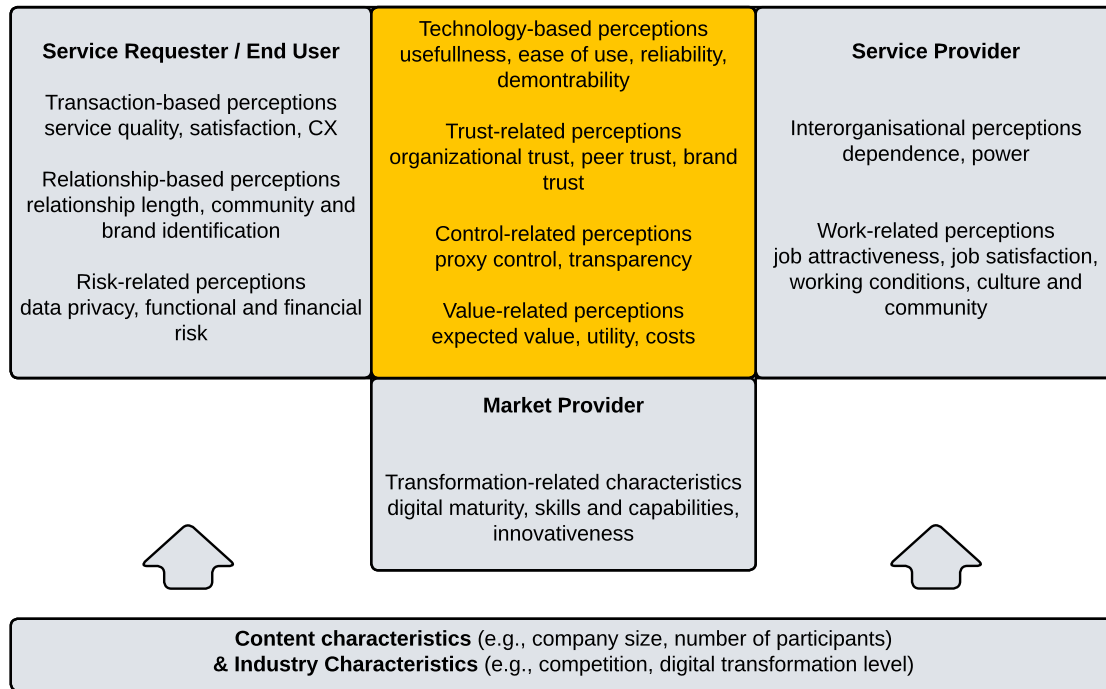
Figure 68: *Framework on multi-stakeholder perspectives on the acceptance of matchmaking platforms.*

Technology perceptions, transcending trust- and control-related beliefs and value perceptions are likely to affect the acceptance of matchmaking platforms for service requesters, service providers and market providers alike. Technology perceptions such as ease of use and perceived usefulness [Dav89] or reliability and result demonstrability have been shown to positively affect the adoption decision of individuals as well as organizational adoption decisions. Trust-related beliefs pertain to trustworthiness perceptions of individuals such as market participants as well as organizational trust in market providers. Trustful relationships have been shown to foster the acceptance of platforms for end consumers and organizations alike. Especially the trust in the brand of the market provider is likely to affect the acceptance of service requesters and service providers. Parallel to trust research, studies on the concept of control have shown that control is a human driving force. Control over technology addresses an individual's or an organization's need to demonstrate competence, superiority, and mastery of technology. Eventually, the value market participants see in using the platform can be considered a main driver for platform acceptance. End requesters might compare the costs and utility of competing platforms to make an informed decision.

In addition to the aforementioned drivers that might pertain to multiple OTF market stakeholders, studies have identified factors that predominantly relate to one stakeholder group. For example, end users'/requesters' acceptance of platforms is likely to be influenced by relational variables beyond trust, such as the length of the relationship between the customer and the platform or individual service providers. Especially in the context of matchmaking platforms with a high level of consociality, a strong platform brand or community identification of end requesters might even outweigh risk perceptions. Research has identified that in technology-mediated service encounters such as transactions on platforms,

risk perceptions (e.g., regarding data privacy, and functional or financial risks) are generally pronounced [PW16]. Although the relationship between the service providers and the market provider does not resemble a professional work relationship, service providers on platforms, who act as solo entrepreneurs, might also count in work-related factors in their decision to use the platform [KBE21]. Interorganizational perceptions such as the service providers' perception of dependence on the market provider, their perception of autonomy, and the power the market provider has over the service providers might influence job satisfaction. Moreover, when an adoption decision is perceived as forced, individuals are likely to have negative emotions and increasing switching intentions. Also, typical work-related perceptions such as job attractiveness, strong relationships, a pleasant working condition, and a friendly work culture and community have been shown to impact employee job satisfaction. We argue that such factors can be considered drivers of platform acceptance, especially if the service providers act as solo entrepreneurs.

A company's decision to extend the pipeline business model or switch to a platform business model and become a market provider is typically dependent on organizational characteristics such as capabilities, the innovativeness of the management and staff, and the overall digital maturity of the firm [WLCH19]. In addition, context factors of the market or network, such as the number of participants, company sizes and industry characteristics such as the competitive level and the level of digitization of the market, are likely to strengthen or weaken the importance of the identified acceptance factors from the perspectives of service requesters, service providers, and market providers.

2.3.3 Application to OTF Computing Markets

We conducted a scenario-based qualitative interview study with 22 potential stakeholders from research institutions as well as companies of a future OTF market. Overall, the study yielded text data from 163 pages of transcribed interview material. Based on the text analysis we could support the importance of most drivers and barriers of our framework (see Figure 68). Most importantly, technology-based perceptions seem to be relevant for all perspectives. For example, one interviewee mentioned the ease of use of the platform as a main driver for the service requesters' acceptance: "*Maybe how it's easy to use that platform. And how you get assistance from . . . our salespeople. . . . Yeah. I think that's the difference.*" (ID 11). Also, value-based perceptions seem crucial for accepting the OTF market from the service requester and service provider perspective as one interviewee mentions: "*You've got to create some value to motivate the users to contribute.*" (ID 5). Brand-related trust as well as community and brand identification were seen as drivers of service requesters' acceptance: "*And of course, it's very important that this platform has a good image. . . . the surrounding communication is very important. . . . What are the comments from users? Is there anything building up? . . . Do you get positive comments from friendly users?*" (ID 6).

Beyond identified factors that apply to the acceptance of individual market participants, we could support the importance of transformation-related factors such as organizational capabilities that influence a company's decision to transform into a market provider. The necessity to develop and maintain a sophisticated skillset of the staff of the market provider is illustrated by the following quote: "*And you also have to think about crucial factors of developing platforms . . . I think you have to have an interdisciplinary team of different*

roles. So, you need, someone who's professional a UX design, you need an IT expert, you need someone who understands how the platform works. . . . I think a big success factor [is] that you've got this mixed team of different roles, different experts, that all work together. Because it's pretty complicated to build a platform from scratch. . . . You have to combine a lot of knowledge" (ID 5).

In sum, we observe that the attraction factors of OTF computing markets are multi-faceted and apply to three perspectives: market provider, service provider, and service requester. Managers of OTF computing markets are encouraged to invest equally in the attraction and retention of both market sides.

3   Impact and Outlook

OTF computing markets are a promising new type of ecosystem to increase the value for potential end users by combining the offering from all involved market participants. In our research, we investigated the underlying enterprise architecture, the development of business models, and acceptance factors for the platform itself as well as the market participants. Our results impact the design of future OTF computing markets as well as the analyzed comparative markets such as mobile ecosystems. For that, first, our developed architectural framework and the SeCoArc tool support software ecosystem designers in the analysis of existing software ecosystems as well as the design of new ones. Second, our BMI4BE method is based on multiple research results from prior research and is an interdisciplinary approach developed by researchers from business administration, information systems, and computer science. By that, the method is one of the first considering multiple research disciplines and was communicated to practice in [RLL+22]. Third, our SBMD tool has been developed for the comparative market of mobile ecosystems. The SBMD is released under open source so that it can be extended by additional features, for example, crowd validation. Fourth, we identified factors that impact the attractiveness of OTF computing markets for multiple stakeholders. These results, for example, that end users' acceptance of platforms is likely to be influenced by the length of the relationship between the customers and the platform provides a solid foundation for further investigations for successful and sustainable OTF computing markets.

In the future, we aim to increase the impact of our contributions by research on simulation and evolution. For that, our first aspect covers dynamics in software ecosystems. Here, we want to understand the evolution of those ecosystems over time with their business, technical, and infrastructure aspects. Second, we want to research the simulation of business models [KV23]. Here, we plan to calculate the financial assessments of business models under different internal and external circumstances. Third, we want to investigate the changes in stakeholder perceptions over time. Here, the aim is to understand changes in the short-term, mid-term, and long-term relationships among the stakeholders.

Acknowledgments

Bibliography

[Adn17]   ADNER, R.: Ecosystem as Structure. In: *Journal of Management* 43 (2017), no. 1, pp. 39–58

[AS16]   AXELSSON, J.; SKOGLUND, M.: Quality assurance in software ecosystems: A systematic literature mapping and research agenda. In: *Journal of Systems and Software* 114 (2016), pp. 69–81

[BB10]   BOSCH, J.; BOSCH-SIJTSEMA, P.: From integration to composition: On the impact of software product lines, global development and ecosystems. In: *Journal of Systems and Software* 83 (2010), no. 1, pp. 67–76

[BKW21]   BEVERUNGEN, D.; KUNDISCH, D.; WÜNDERLICH, N.: Transforming into a platform provider: strategic options for industrial smart service providers. In: *Journal of Service Management* 32 (2021), no. 4, pp. 507–532

[Dav89]   DAVIS, F. D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: *MIS Quarterly* 13 (1989), no. 3, p. 319

[GA03]   GORDIJN, J.; AKKERMANS, J. M.: Value-based requirements engineering: exploring innovative e-commerce ideas. In: *Requirements Engineering* 8 (2003), no. 2, pp. 114–134

[GYNE21]   GOTTSCHALK, S.; YIGITBAS, E.; NOWOSAD, A.; ENGELS, G.: Situation-Specific Business Model Development Methods for Mobile App Developers. In: *Enterprise, Business-Process and Information Systems Modeling*. Ed. by AUGUSTO, A.; GILL, A.; NURCAN, S.; REINHARTZ-BERGER, I.; SCHMIDT, R.; ZDRAVKOVIC, J. Vol. 421. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2021, pp. 262–276

[GYNE22a]   GOTTSCHALK, S.; YIGITBAS, E.; NOWOSAD, A.; ENGELS, G.: Continuous situation-specific development of business models: knowledge provision, method composition, and method enactment. In: *Software and Systems Modeling* (2022), no. 22, pp. 47–73

[GYNE22b]   GOTTSCHALK, S.; YIGITBAS, E.; NOWOSAD, A.; ENGELS, G.: Situational Business Model Developer: A Tool-support for Situation-specific Business Model Development. In: *Wirtschaftsinformatik 2022 Proceedings*. AIS, 2022

[GYNE22c]   GOTTSCHALK, S.; YIGITBAS, E.; NOWOSAD, A.; ENGELS, G.: Towards Software Support for Situation-Specific Cross-Organizational Design Thinking Processes. In: IWSiB '22. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2022, pp. 1–8.

[HRÅR14]   HENDERSON-SELLERS, B.; RALYTÉ, J.; ÅGERFALK, P. J.; ROSSI, M.: *Situational Method Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014

[JCG18]   JACOBIDES, M. G.; CENNAMO, C.; GAWER, A.: Towards a theory of ecosystems. In: *Strategic Management Journal* 39 (2018), no. 8, pp. 2255–2276

[JPEK16]   JAZAYERI, B.; PLATENIUS, M. C.; ENGELS, G.; KUNDISCH, D.: Features of IT Service Markets: A Systematic Literature Review. In: *Service-Oriented Computing*. Ed. by SHENG, Q. Z.; STROULIA, E.; TATA, S.; BHIRI, S. Vol. 9936. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 301–316

[JPML21]   JUNG, J.; PARK, E.; MOON, J.; LEE, W. S.: Exploration of Sharing Accommodation Platform Airbnb Using an Extended Technology Acceptance Model. In: *Sustainability* 13 (2021), no. 3, p. 1185

[JZEK17]   JAZAYERI, B.; ZIMMERMANN, O.; ENGELS, G.; KUNDISCH, D.: A Variability Model for Store-Oriented Software Ecosystems: An Enterprise Perspective. In: *Service-Oriented Computing*. Ed. by MAXIMILIEN, M.; VALLECILLO, A.; WANG, J.; ORIOL, M. Vol. 10601. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 573–588

[JZK+18]   JAZAYERI, B.; ZIMMERMANN, O.; KÜSTER, J.; ENGELS, G.; SZOPINSKI, D.; KUNDISCH, D.: Patterns of Store-oriented Software Ecosystems: Detection, Classification, and Analysis of Design Options. In: *Proceedings of Latin American Conference on Pattern Languages of Programming*. 2018

[KBE21]   KARANOVIĆ, J.; BERENDS, H.; ENGEL, Y.: Regulated Dependence: Platform Workers' Responses to New Forms of Organizing. In: *Journal of Management Studies* 58 (2021), no. 4, pp. 1070–1106

[KV23]   KSOURI-GERWIEN, C.; VORBOHLE, C.: Supporting Business Model Decision-making in B2B Ecosystems: A Framework for Using System Dynamics. In: *Proceedings of the 55th Hawaii International Conference on System Sciences (HICSS)*. 2023

[OP10]   OSTERWALDER, A.; PIGNEUR, Y.: *Business model generation: a handbook for visionaries, game changers, and challengers*. Hoboken, New Jersey: John Wiley & Sons, 2010

[PFG13]   PERNSTÅL, J.; FELDT, R.; GORSCHEK, T.: The lean gap: A review of lean approaches to large-scale software systems development. In: *Journal of Systems and Software* 86 (2013), no. 11, pp. 2797–2821

[PK18]   PERREN, R.; KOZINETS, R. V.: Lateral Exchange Markets: How Social Platforms Operate in a Networked Economy. In: *Journal of Marketing* 82 (2018), no. 1, pp. 20–36

[PW16]   PALUCH, S.; WÜNDERLICH, N. V.: Contrasting risk perceptions of technology-based service innovations in inter-organizational settings. In: *Journal of Business Research* 69 (2016), no. 7, pp. 2424–2431

[RLL+22]   ROBRA-BISSANTZ, S.; LATTEMANN, C.; LAUE, R.; LEONHARD-PFLEGER, R.; WAGNER, L.; GERUNDT, O.; SCHLIMBACH, R.; BAUMANN, S.; VORBOHLE, C.; GOTTSCHALK, S.; KUNDISCH, D.; ENGELS, G.; WÜNDERLICH, N.; NISSEN, V.; LOHRENZ, L.; MICHALKE, S.: Methoden zum Design digitaler Plattformen, Geschäftsmodelle und Service-Ökosysteme. In: *HMD Praxis der Wirtschaftsinformatik* 59 (2022), no. 5, pp. 1227–1257

[SE20]   SCHWICHTENBERG, B.; ENGELS, G.: SecoArc: A Framework for Architecting Healthy Software Ecosystems. In: *Software Architecture*. Ed. by MUCCINI, H.; AVGERIOU, P.; BUHNOVA, B.; CAMARA, J.; CAPORUSCIO, M.; FRANZAGO, M.; KOZIOLEK, A.; SCANDURRA, P.; TRUBIANI, C.; WEYNS, D.; ZDUN, U. Vol. 1269. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, pp. 95–106

[SL20]   SCHWARZ, J. S.; LEGNER, C.: Business model tools at the boundary: exploring communities of practice and knowledge boundaries in business model innovation. In: *Electronic Markets* 30 (2020), no. 3, pp. 421–445

[SMJ+22]   SZOPINSKI, D.; MASSA, L.; JOHN, T.; KUNDISCH, D.; TUCCI, C. L.: Modeling Business Models: A cross-disciplinary Analysis of Business Model Modeling Languages and Directions for Future Research. In: *Communications of the Association for Information Systems* (2022), no. 51, pp–pp

[SSJ+20]   SZOPINSKI, D.; SCHOORMANN, T.; JOHN, T.; KNACKSTEDT, R.; KUNDISCH, D.: Software tools for business model innovation: current state and future challenges. In: *Electronic Markets* 30 (2020), no. 3, pp. 469–494

[Tee10]   TEECE, D. J.: Business Models, Business Strategy and Innovation. In: *Long Range Planning* 43 (2010), no. 2/3, pp. 172–194.

[VK22]   VORBOHLE, C.; KUNDISCH, D.: Overcoming Silos: A Review of Business Model Modeling Languages for Business Ecosystems. In: *Proceedings of the 30th European Conference on Information Systems (ECIS), Research-in-Progress*. 2022

[WLCH19]   WILLIAMS, P. A.; LOVELOCK, B.; CABARRUS, T.; HARVEY, M.: Improving Digital Hospital Transformation: Development of an Outcomes-Based Infrastructure Maturity Assessment Framework. In: *JMIR medical informatics* 7 (2019), no. 1

[ZA13]   ZOTT, C.; AMIT, R.: The business model: A theoretically anchored robust construct for strategic analysis. In: *Strategic Organization* 11 (2013), no. 4, pp. 403–411

Transfer Project T1:

Flexible Industrial Analytics on Reconfigurable Systems-On-Chip

Alexander Boschmann[2], Lennart Clausing[1], Felix Jentzsch[1], Hassan
Ghasemzadeh Mohammadi[1], Marco Platzner[1]

1 Department of Computer Science, Paderborn University,
Paderborn, Germany

2 Weidmüller Interface GmbH & Co. KG., Germany

1 Introduction

Industrial analytics refers to the current trend in automation technology to collect and analyze a variety of measured values from machines and from production processes in order to generate added value for future operations. Industrial analytics is a business field with great economic potential, and Weidmüller wants to position itself as a leading provider of industrial analytics solutions within the framework of the mission statement *Industry 4.0*[24]. Examples for industrial analytics include the detection of significant deviations from the target behavior of a machine [MKPN13], the detection of inefficiencies [MPK15], the creation of fault forecasts and the diagnosis of fault causes. The added value achieved is the avoidance of machine breakdowns, the minimization of downtime, or in general, the increase of plant productivity and production output [PKG+16].

In *embedded analytics*, i.e., the implementation of analysis functions directly in the automation devices within a production plant, Weidmüller relies on reconfigurable System-on-Chip (rSoC). The challenges in using rSoC for industrial analytics are on the one hand the required *flexibility in system design* and, on the other hand, the increasing *heterogeneity of rSoC platforms*. Flexibility is needed since the functions of industrial analytics have to be selected, configured and assembled on an application-specific basis, implemented as a hardware/software co-design and deployed on an rSoC. Flexibility can further be exploited during runtime to use the resources efficiently under varying load situations. The technological evolution of rSoC platforms is toward more heterogeneous architectures: for example, recent rSoCs combine multiple processor types with reconfigurable hardware, embedded graphics processors, and application-specific blocks.

The combination of increasing *dynamics* of tasks and *heterogeneity* of the underlying architectures is also the guiding theme of basic scientific research in subproject C2 of SBF 901. There, novel architectures and programming models for heterogeneous computing nodes are investigated and developed. By transferring these basic scientific results to the industrial analytics application domain, this transfer project aims to achieve the following goals:

Alexander.Boschmann@weidmueller.com (Alexander Boschmann), lennart.clausing@upb.de (Lennart Clausing), felix.jentzsch@upb.de (Felix Jentzsch), ghasemzadeh@gmail.com (Hassan Ghasemzadeh Mohammadi), platzner@ubp.de (Marco Platzner)

[24]`https://www.weidmueller.com/int/solutions/industrial_analytics/index.jsp`

1. Characterization of essential functions of industrial analytics and design of hardware/software partitionings suitable for an rSoC implementation.

2. Development of architectures and programming environments to enable transmodal migration on rSoC.

3. Demonstration of rSoC technology for industrial analytics use cases.

For rSoC architectures and programming environments we draw on preliminary work, the ReconOS [AHK+14] operating system for reconfigurable computers. ReconOS allows for multithreaded programming across the software/hardware boundary by turning accelerator functions into so-called hardware threads and semantically integrating them as threads into a guest operating system environment. Compared to related approaches such as BOPRH [KB08], Hthreads [ASA+08], FUSE [IS11], SPREAD [WZW+13], and LEAP [FYAE14], ReconOS is more flexible and more rapidly portable to new guest operating systems and FPGA technologies. In particular, ReconOS has demonstrated its usefulness in three scenarios: First, ReconOS supports a step-by-step development flow starting from a software application prototype on desktop under Linux. Only when the prototype is functionally correct, is the application ported to the embedded rSoC, which is typically a low effort since the embedded CPU cores also run Linux. As a last step, threads that are amenable to hardware acceleration are gradually moved from software to hardware. Second, ReconOS facilitates design space exploration since different hardware/software partitionings can easily be generated by simply modifying system configuration data and no changes are needed to unaffected threads or the operating system. This feature has been used, for example, to develop a video object tracking system [HLP13]. Finally, ReconOS even allows for the construction of adaptive or self-adaptive systems, where a hardware/software application monitors its own performance and changes the architecture, for example the number of used CPU cores and hardware threads or the hardware/software partitioning, in reaction to a varying workload [AHL+14].

2 Main Contributions

In the course of the transfer project, we have achieved the following results:

- We have developed ReconOS^{64} as a new version of the ReconOS architecture and operating system layer for the modern 64-bit rSoC technology used in the project, i.e., the Xilinx UltraScale+ MPSoC platform FPGAs [CP22; Cla21].

- We have created a build tool flow for ReconOS^{64} that includes a high-level synthesis (HLS) tool flow and thus allows for creating hardware threads not only in hardware description languages such as VHDL and Verilog, but also in C/C++.

- We have implemented several industrial analytics functions as software/hardware co-designs on the rSoC platform, including k-NN [Ria17], decison trees/random forests, SVM [BTW+17], and neural network models [Nga22].

- We have worked on several industrial analytics case studies for condition monitoring and anomaly detection, respectively, targeting wind turbines, molding machines and welding machines [Kau22].

In the following, we select two of these topics for elaboration, the ReconOS^{64} development and the DEEPWIND case study, a condition monitoring system for wind turbines.

2.1 The ReconOS[64] Operating System for 64-bit Platform FPGAs

ReconOS[64] bases on previous ReconOS [AHK+14; LP09] implementations but targets modern platform FPGAs with 64-bit processors. The step towards 64-bit support and the use of modern platform FPGAs is important, since many applications, in particular industrial analytics functions, require the increased computing capabilities and resources provided by modern rSoC. ReconOS and its 64-bit version are freely available in open source[25].

Figure 69 shows the architecture of ReconOS[64] on the Xilinx UltraScale+ MPSoC. The processing system (PS) comprises a 64-bit quad-core CPU and runs the 64-bit Xilinx PetaLinux as the host operating system. ReconOS[64] extends the host operating system by the ReconOS driver in kernel space and several libraries in user space for, e.g., thread synchronization, communication, management and bitstream loading. The programmable logic part of the platform FPGA is structured into so-called reconfigurable slots that constitute rectangular areas of the programmable logic fabric. Reconfigurable slots accommodate hardware threads, which are ReconOS' abstractions of accelerated functions. A main feature of ReconOS is that hardware threads access the operating system using the same services as software threads running on the CPU, thus enabling the multithreaded programming abstraction across the hardware/software boundary. This is made possible by delegate threads, light-weight software threads that conduct operating system calls on behalf of their corresponding hardware threads.
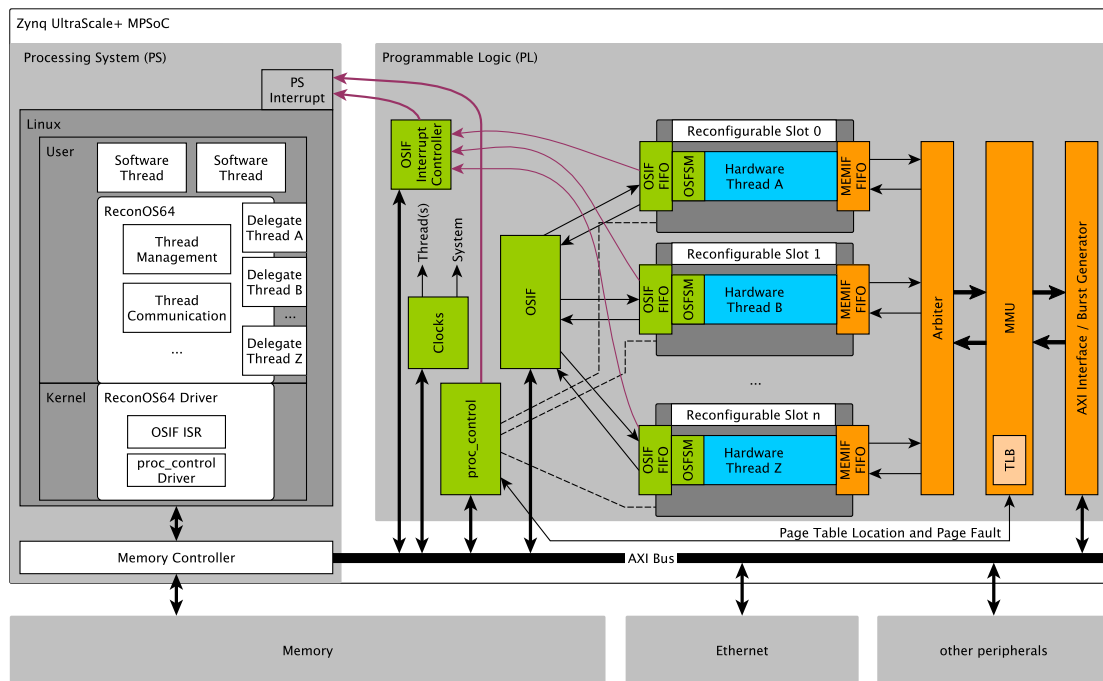

Figure 69: *ReconOS[64] Architecture on the Xilinx UltraScale+ MPSoC (taken from [CP22]).*

The intellectual property (IP) cores of ReconOS[64] responsible for connecting hardware threads with the host operating system are shown in green color in Figure 69. Each

---

[25]www.reconos.de

hardware thread comprises the actual user logic and an operating system finite state machine (OSFSM) that sequentializes the thread's operating system interactions and handles synchronization between the user logic and the software. Further, each hardware thread is connected to an operating system interface (OSIF) FIFO that buffers the operating system calls, i.e., their commands with parameters and return values. The FIFO also serves to separate the clock regions of the ReconOS^{64} design from the hardware threads to allow them to run at different frequencies. The OSIF FIFOs connect to a central OSIF IP core that collects the commands and parameters for all service requests and, in addition, to a dedicated OSIF interrupt controller that raises a CPU interrupt whenever a hardware thread wants to execute an operating system call. On the software side, the raised interrupt will activate the OSIF interrupt service routine (ISR), which in turn sets the delegate thread corresponding to the hardware thread that is ready to run. The delegate thread then accesses the OSIF IP core, retrieves the command and parameters and actually performs the operating system call. In case there are return values, they are written back to the hardware threads.

The proc_control IP core together with the proc_control kernel driver are also involved in operating system communication as they propagate reset signals towards the hardware threads. In ReconOS^{64} the native data type is 64 bit. Hence, all IP cores involved in operating system communication support command, parameter, and return data structures in multiples of 64-bit. This is particularly important since many operating system calls, e.g., message box reads and writes, are typically used to communicate 64-bit pointers between software and hardware threads. A consequence of the 64-bit orientation is that data of smaller width has to be either padded to the next multiple of eight bytes or, if several small-sized data are to be written or read, concatenated to blocks of eight byte.

The IP cores of ReconOS^{64} responsible for supporting memory accesses of the hardware threads are displayed in orange color in Figure 69. While address pointers in ReconOS^{64} are 64-bit wide, the Linux configuration we use on the ARMv8 CPU architecture uses a virtual address space of 512 GB that is mapped to a physical address space of 256 TB. Hence, the systems' memory management unit (MMU) considers only the lower 39 bit of virtual addresses and deals with 48-bit physical addresses. The page size in our configuration amounts to 4 KB. Hardware threads use virtual addresses and access memory via their memory interfaces (MEMIF). ReconOS^{64} employs three IP cores for establishing memory access. The Arbiter resolves simultaneous accesses from different hardware threads, the MMU performs the translation to physical addresses, and the AXI Interface / Burst Generator interfaces to the AXI bus and ensures burst transfers. Initially, the content of the ARM CPU's Translation Table Base Register (TTBR) is transferred to the MMU via the kernel driver and the proc_control IP core to ensure that the MMU has the physical address of the ReconOS process' first-level page table. Then, the MMU performs the page table walk which results in at most three memory accesses. To speed up memory access for hardware threads, the MMU includes a translation look-aside buffer (TLB) that caches recent translations between the 27-bit virtual page numbers and the 36 bit physical page numbers. The size of the TLB is configurable. The proc_control component supports the handling of page faults during address translation. Therefore, proc_control needs to be able to raise an interrupt with the CPU.

Dynamic thread management is supported through partial reconfiguration in ReconOS^{64}. Generally, a hardware thread is assigned to a reconfigurable slot, which is a rectangular

area of logic resources on the FPGA residing in the same clock region. A new feature of ReconOS[64] are *reconfigurable slot groups*, which specify sets of reconfigurable slots of the same size. Each hardware thread is assigned to one or more such reconfigurable slot groups, and multiple hardware threads can be assigned to the same reconfigurable slot group. The introduction of reconfigurable slot groups makes the runtime mapping between hardware threads and reconfigurable slots more flexible.

ReconOS[64] allows for the hardware threads to run at individual clock rates, in particular different ones from the clock of the static ReconOS part. These individual clock signals are fed from the ReconOS[64] clocking IP core that utilizes a Mixed-mode Clock Manager (MMCM) tile with static multiplier and variable dividers for each clock output. Reconfigurable slot groups can be assigned to individual clocks as long as the clock tile resources are not exceeded. Using a function from the ReconOS[64] thread management library, both software and hardware threads can set the clock frequencies for hardware threads by modifying the clock dividers in the corresponding ReconOS[64] clocking IP core.
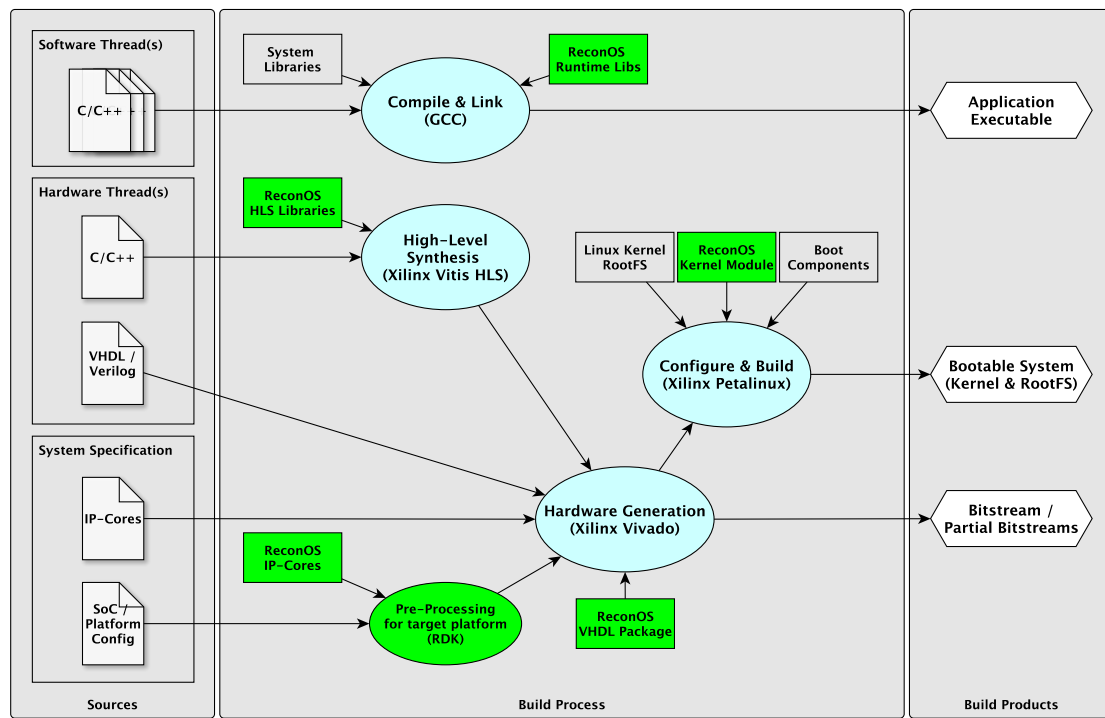

Figure 70: *ReconOS[64] build tool flow (adapted from [AHK+14]).*

The ReconOS[64] build tool flow takes as inputs the sources for the application's software threads in C/C++ and the sources for the hardware threads in either VHDL/Verilog or C/C++ for use with high-level synthesis (HLS). For both, a predefined set of ReconOS[64] - specific library functions is provided. A further input is the system specification comprising a set of ReconOS[64] IP cores and the configuration file. The configuration file includes definitions for the target platform, the reconfigurable slots, and reconfigurable slot groups and assigns the hardware threads to reconfigurable slot groups. Further, all operating system service objects, such as message boxes, mutexes, and semaphores, are listed in the configuration file. The build process relies on a Python-based templating system. The application software is cross-compiled with the `aarch64-gcc` compiler, which results in

the application executable. On the hardware side of the build tool flow, the ReconOS Development Kit (RDK) processes the configuration file and generates IP sources from architecture- and board-specific templates. The flow then generates a Xilinx Vivado project incorporating the user-provided hardware threads, either directly from the VHDL/Verilog code or the result from HLS, with all necessary components and connections. The hardware build process results in the static bitstream for the ReconOS[64] system and a set of partial bitstreams for the hardware threads. Information from the hardware build process (e.g., used address ranges for IP cores) together with a generic Xilinx PetaLinux template project, the ReconOS[64] kernel module and device tree, and boot components are used to configure and generate a bootable system including the operating system kernel and the root file system.

2.2 DeepWind: An Accurate Wind Turbine Condition Monitoring Framework via Deep Learning on Embedded Platforms

The generation of electricity using wind turbines is rapidly growing and becoming more important since it is considered as an affordable and clean substitute for fossil fuel-based electricity production. Wind turbines are used in a large variety of environments, both onshore and offshore, and they are exposed to harsh working conditions, such as unbalanced wind load, wind turbulence, and large temperature variations [QL15]. To service running wind turbines unceasingly and safely, and particularly reduce the maintenance costs, adequate online *condition monitoring systems* (CMSs) are required [Wei]. CMSs identify the type and the location of faults and, more importantly, diagnose the transformation of a fault into an error and possibly into a failure.

During wind turbine operation, a CMS constantly takes measurements that determine the condition of the critical components, e.g., the rotor blades. The measurements provide indications for problems such as blade damages after lightning strikes, heavy vibrations of the blades, or the ice accretion on a rotor blade. Ice accretion may lead to dangerous ice throw, which is a major risk for the surroundings of wind turbines. Therefore, more and more local authorities insist on blade measuring ice detection systems. By processing the information of a CMS, a diagnosis, e.g., *inspection*, *necessary repair*, or *necessity of turbine shutdown*, is reached and an adequate maintenance plan is formulated. Consequently, the CMS facilitates low-cost maintenance before a critical failure happens while diminishing the downtime of the wind turbine, also increasing its dependability and lifetime. Defects can cause abnormal vibrations of the blades, which can be sensed by accelerometers installed on the blades. Earlier work applied frequency spectrum analyses [CG05; WX06] to detect such defects. However, such analyses require manual feature engineering and extensive trial-and-error to identify patterns in the vibrations that correctly match to faulty cases.

In DeepWind, we propose a novel framework to build an end-to-end condition monitoring and fault detection system for rotor blades. The framework starts with a preprocessing step to reduce the complexity of the raw sensor data. Then, inspired by the success of deep learning in time series analysis, we train a multi-channel convolutional neural network (MC-CNN) that can automatically extract a set of discriminative features from the sensor data. Finally, the trained MC-CNN is automatically mapped to an embedded

FPGA platform, where a combination of software and hardware identifies fault occurrences within the data streamed from accelerometer sensors.
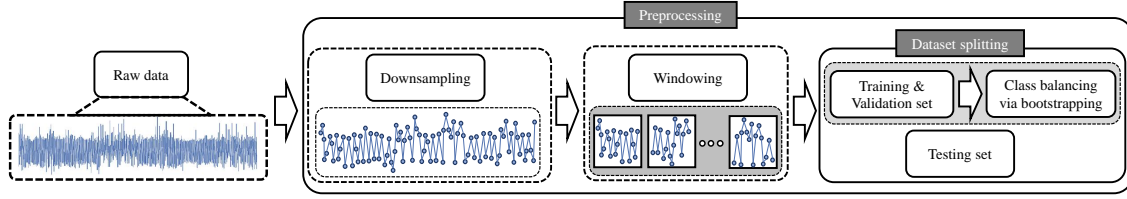

Figure 71: *Main steps of data preprocessing: downsampling, windowing, data set splitting, and training set bootstrapping (taken from [GAR⁺20]).*

Figure 71 illustrates the main steps of the sensor data preprocessing. The main goal of the preprocessing step is to convert the raw sensor data into a cleanly formatted data set that can be used later by the MC-CNN for fault detection purposes. The complexity of sensor data is reduced by applying downsampling, in which the number of sample points in the input data is reduced. We utilize the *Mode-Median-Bucket* algorithm [Ste13], in which every window is divided into several subwindows in such a way that each subwindow contains the same number of samples. The algorithm considers important features from each bucket with mode, median, global peaks, or global trough values and filters out the other samples in each subwindow. In the next step, we utilize the windowing technique to divide each input data frame into a number of smaller segments called windows. Each window simply adopts the label of its data frame. Finally, we form our training and testing data sets from the obtained windows. We modify the training set by bootstrapping with replacement to ensure that the number of samples from both faulty and non-faulty classes are comparable. This is an important step to be able to train a high-quality classifier that provides high accuracy and recall on the testing set [Man22].

As the target feature extractor and classifier, we exploit a multi-channel CNN, in which the training of each individual univariate data, e.g., raw data from each sensor, is performed independently [Kau22]. Indeed, we can draw a lot of inference from the local properties of each sensor without losing the generality of our classifier, by decoupling the data of different sensors. The architecture of the MC-CNN model we have used in this work is shown in Figure 72. After preprocessing the sensor data we apply a Fast Fourier Transform (FFT) on each input window to extract its Spectrum Frequency (SF). The obtained SFs are then fed into the MC-CNN.

The first part of the MC-CNN performs feature extraction and contains two 1-D convolution layers as well as two max-pooling layers. For each sensor, the so-called channel, we utilize 50 and 40 feature maps in the first and second convolution layers with the size of 8 and 4, respectively. As we have two sensors per blade, we exploit six 1-D convolution channels. The outcomes of each convolution layer are downsampled by a max-pooling layer to control the growth in the size of the extracted features. Finally, the obtained features are fed into a fully connected layer with 400 neurons. This layer is followed by a softmax layer that generates the conditional probability of faulty and non-faulty classes. Note that the training of the MC-CNN is performed offline, and then the trained model is quantized and mapped on the hardware for the inference phase.

On the hardware side of our framework, called TFPGA, we utilize an rSoC as the target platform. We exploit hardware/software codesign to both efficiently distribute the
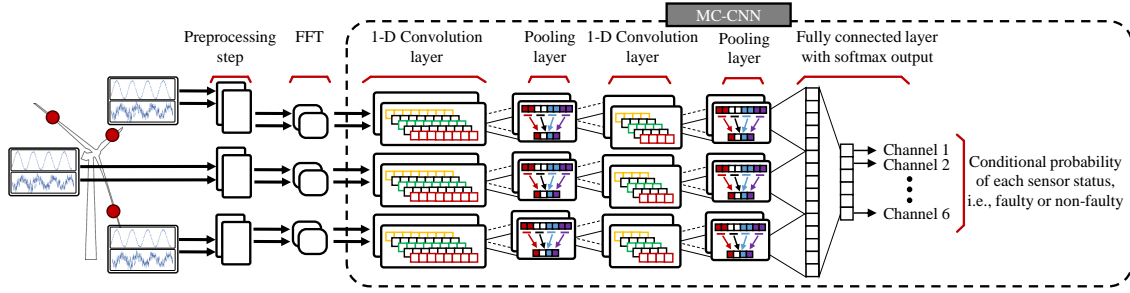
Figure 72: *The architecture of multi-channel CNN for fault detection. The convolution layers have 50 and 40 filters with kernel sizes of 8 and 4, respectively. For each window, obtained from the preprocessing step, the FFT spectrum is computed and the outcome is fed to the MC-CNN (taken from [GAR⁺20]).*

framework tasks on various rSoC resources and benefit from the customizability and parallelism offered by FPGAs. For inference, the tasks of the preprocessing step, e.g., downsampling and windowing, as well as the FFT computation for each obtained window are assigned to the CPUs of the rSoC. Note that the bootstrapping task is just performed on the training data set and is omitted from consideration in the inference phase. Next, the trained MC-CNN network is analyzed, and a C++ implementation of this model is created. This code is then given to Xilinx SDSoC to create a bitstream needed for the target FPGA. To improve the execution time of the software model and reduce its size, the framework exploits a custom precision scaling feature that enables a designer to utilize the underlying hardware more efficiently by tuning the parameters of the given network.

We have used a real-word data set provided by Weidmüller Monitoring Systems GmbH to evaluate the approach. The data set comprises time series data measured with a sampling rate of 1 kHz from the edge-wise and flap-wise sensors for each of three rotor blades. For every half hour (i.e., 1.8 million sample points), the data is labeled with the sensor status in that interval as *faulty* or *non-faulty*. After preprocessing, we have obtained samples with a window size of 1 second along with corresponding labels. We have used 80% of the data for training and validation and the remaining 20% for testing. All the models have been implemented with the Keras library [Ker].

Figure 73 shows the classification result of our DEEPWIND framework. The figure plots the achieved F1-score versus the six channels, i.e., two per blade, where each channel represents a sensor blade. The F1-score is the harmonic mean of precision and recall and reaches its best value at 1, which translates to perfect precision and recall. Our proposed MC-CNN based fault detection scheme provides an average of 0.94 for the F1-score. As a baseline technique, we have experimented with a Support Vector Machine (SVM) used previously by the application partner, which results in an F1-score of 0.64 on average. Importantly, for all of the six channels MC-CNN provides better classification results in comparison with SVM, making our MC-CNN approach a successful technique to capture the most discriminative features for the sensor blade fault detection problem.

Figure 74 represents the accuracy for various quantization settings and for the reference software implementation, which utilizes double-precision floating point. The results show an accuracy penalty of 6% for a 16-bit quantization, which we deem acceptable without model retraining. When only the weights are quantized further to 8 bit, we even observe

a slight increase in accuracy to 87%. We attribute this to the inherent regularization characteristic of the quantization, since we know that the original model benefits from dedicated regularization, namely through dropout. We have also measured the resource usage for different quantization settings. The initial results revealed that going from 16 to 8 bit leads to a slight saving in lookup tables of 12%. Quantization of weights to 8 bit achieves a 34% decrease in embedded memory (BRAM) usage. These results show that weight quantization is effective for reducing the memory footprint in an MC-CNN hardware accelerator.
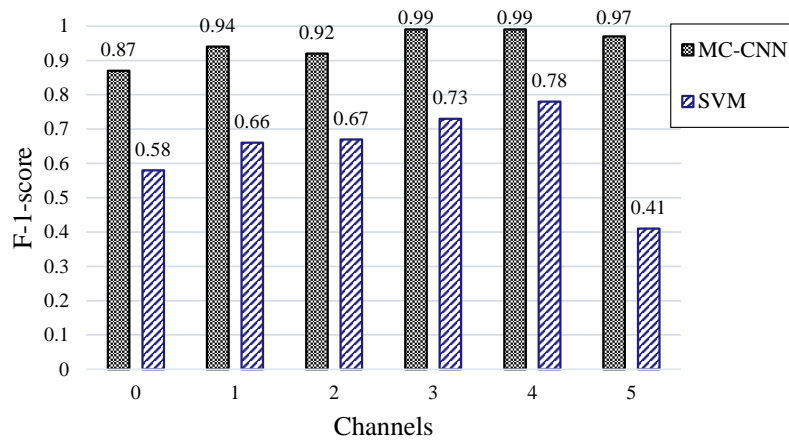

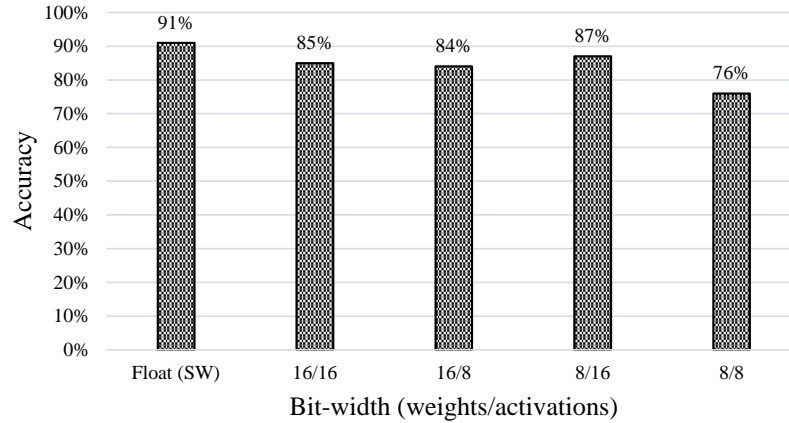Figure 73: *F1-score of MC-CNN and SVM methods for six channels (taken from [GAR+20]).*

Figure 74: *Accuracy vs. bit width of the MC-CNN on a Xilinx UltraScale+ MPSoC (taken from [GAR+20]).*

3 Impact and Outlook

This transfer project allowed us to further develop and apply reconfigurable system-on-chip technology to the concrete application domain of industrial analytics functions, together with the application partner Weidmüller. Overall, the project was successful since the newly developed 64-bit ReconOS[64] architecture provides CPU cores with sufficient compute power and hardware acceleration for industrial analytics functions. The corresponding

build tool flow has shown to support the specification of runtime-reconfigurable functions in a rather simple way. We have implemented a number of typical functions of industrial analytics as software/hardware co-designs with ReconOS[64] and demonstrated that different trade-offs between performance and resource consumption can be explored. Our developments are open source and can thus be used and leveraged by others. We have also worked on several use cases, where condition monitoring for wind turbines is so far the most successful one. For this use case, we could propose an industrial analytics function that greatly improves the existing solution in terms of quality. The mapping to an embedded rSoC is also of great interest, since then the condition monitoring system can be placed near the sensors in the rotor blades and running such functions on servers in wind turbines can be avoided.

One aspect planned for this transfer project could not yet be realized in a use case, the transmodal migration of industrial analytics functions. While this feature has been demonstrated in the lab, for the concrete use cases it was more important to spend time for developing industrial analytics functions that excel in functional quality. One particular challenge for developing good solutions is that often only small data sets or data with very imbalanced classes are available.

Ongoing and future work includes the development of more solutions for condition monitoring and predictive maintenance, in particular for welding machines [Kum23], and the further optimization of ReconOS[64].

For mapping DNN architectures to rSoC, in this transfer project we have first used our TFPGA framework (cf. Section 2.2) and, later, we have developed a framework that focuses on TF Lite[26] with its backend delegate modules. We have developed an FPGA delegate for TF Lite that facilitates the necessary hardware/software co-design using the ReconOS[64] architecture and operating system (cf. Section 2.1). The partial reconfiguration support of ReconOS[64] enables the instantiation of model-tailored accelerator architectures. Mapping DNNs to rSoC technology remains an area of active research. Recently, we have switched to the open source FINN framework [BPF+18] that maps DNNs as streaming dataflow architectures to FPGAs and features flexible quantization as well as quantization-aware DNN training.

Bibliography

[AHK+14]   Agne, A.; Happe, M.; Keller, A.; Lubbers, E.; Plattner, B.; Platzner, M.; Plessl, C.: ReconOS: An Operating System Approach for Reconfigurable Computing. In: *IEEE Micro* 34 (Jan. 2014), no. 1, pp. 60–71

[AHL+14]   Agne, A.; Happe, M.; Lösch, A.; Plessl, C.; Platzner, M.: Self-awareness as a Model for Designing and Operating Heterogeneous Multicores. In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7 (2014), no. 213

[ASA+08]   Andrews, D.; Sass, R.; Anderson, E.; Agron, J.; Peck, W.; Stevens, J.; Baijot, F.; Komp, E.: Achieving Programming Model Abstractions for Reconfigurable Computing. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16 (2008), no. 1, pp. 34–44

[BPF+18]   Blott, M.; Preusser, T. B.; Fraser, N. J.; Gambardella, G.; O'brien, K.; Umuroglu, Y.; Leeser, M.; Vissers, K.: FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks. In: 11 (2018), no. 3.

---

[26]https://www.tensorflow.org/lite

[BTW+17]   BOSCHMANN, A.; THOMBANSEN, G.; WITSCHEN, L.; WIENS, A.; PLATZNER, M.: A Zynq-based
            dynamically reconfigurable high density myoelectric prosthesis controller. In: *In Proceed-
            ings of Design, Automation and Test in Europe (DATE)*. IEEE, 2017

[CG05]   CASELITZ, P.; GIEBHARDT, J.: Rotor condition monitoring for improved operational safety of
            offshore wind energy converters. In: *J. Sol. Energy Eng.* 127 (2005), no. 2, pp. 253–261

[Cla21]   CLAUSING, L.: ReconOS64: High-Performance Embedded Computing for Industrial An-
            alytics on a Reconfigurable System-on-Chip. In: *Proceedings of the 11th International
            Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*. ACM,
            2021

[CP22]   CLAUSING, L.; PLATZNER, M.: ReconOS64: A Hardware Operating System for Modern Plat-
            form FPGAs with 64-Bit Support. In: *2022 IEEE International Parallel and Distributed
            Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 120–127

[FYAE14]   FLEMING, K.; YANG, H.-J.; ADLER, M.; EMER, J.: The LEAP FPGA operating system. In:
            International Conference on Field Programmable Logic and Applications (FPL). IEEE,
            2014

[GAR+20]   GHASEMZADEH MOHAMMADI, H.; ARSHAD, R.; RAUTMARE, S.; MANJUNATHA, S.; KUSCHEL, M.;
            JENTZSCH, F. P.; PLATZNER, M.; BOSCHMANN, A.; SCHOLLBACH, D.: DeepWind: An Accu-
            rate Wind Turbine Condition Monitoring Framework via Deep Learning on Embedded
            Platforms. In: *2020 25th IEEE International Conference on Emerging Technologies and
            Factory Automation (ETFA)*. 2020

[HLP13]   HAPPE, M.; LÜBBERS, E.; PLATZNER, M.: A Self-adaptive Heterogeneous Multi-core Ar-
            chitecture for Embedded Real-time Video Object Tracking. In: *International Journal of
            Real-time Image Processing* 8 (2013), no. 1, pp. 95–110

[IS11]   ISMAIL, A.; SHANNON, L.: FUSE: Front-End User Framework for O/S Abstraction of
            Hardware Accelerators. In: *International Symposium on Field-Programmable Custom
            Computing Machines (FCCM)*. IEEE, 2011

[Kau22]   KAUR, P.: Analysis of Time-Series Classification in Conditional Monitoring Systems.
            MA thesis. Paderborn University, 2022

[KB08]   KWOK-HAY SO, H.; BRODERSEN, R.: Runtime Filesystem Support for Reconfigurable FPGA
            Hardware Processes in BORPH. In: *International Symposium on Field-Programmable
            Custom Computing Machines (FCCM)*. 2008

[Ker]   KERAS: *The Python Deep Learning API*. https://keras.io

[Kum23]   KUMAR, N. Y. M.: Data Analytics for Predictive Maintenance of Time Series Data. MA
            thesis. Paderborn University, 2023

[LP09]   LÜBBERS, E.; PLATZNER, M.: ReconOS: Multithreaded Programming for Reconfigurable
            Computers. In: *ACM Trans. Embed. Comput. Syst.* 9 (Oct. 2009), no. 1

[Man22]   MANJUNATHA, S.: Dealing with Pre-Processing and Feature Extraction of Time-Series Data
            in Predictive Maintenance. MA thesis. Paderborn University, 2022

[MKPN13]   MAIER, A.; KÖSTER, M.; PAIZ GATICA, C.; NIGGEMANN, O.: Automated Generation of Timing
            Models in Distributed Production Plants. In: *Proceedings of the IEEE International
            Conference on Industrial Technology (ICIT)*. IEEE, 2013

[MPK15]   MICHELS, J. S.; PAIZ GATICA, C.; KÖSTER, M.: Anomalien und Ineffizienz in Produktionsan-
            lagen erkennen. In: *atp edition - Automatisierungstechnische Praxis* 57 (2015), no. 10,
            p. 26

[Nga22]   NGAYAP, V. I. T.: FreeRTOS on a MicroBlaze Soft-Core Processor with Hardware Acceler-
            ators. MA thesis. Paderborn University, 2022

[PKG+16]   PAIZ GATICA, C.; KÖSTER, M.; GAUKSTERN, T.; BERLIN, E.; MEYER, M.: An Industrial Analyt-
            ics Approach to Predictive-Maintenance for Machinery Applications. In: *Proceedings of
            the IEEE International Conference on Emerging Technologies and Factory Automation*.
            2016

[QL15]   Qiao, W.; Lu, D.: A survey on wind turbine condition monitoring and fault diagnosis—Part I: Components and subsystems. In: *IEEE Transactions on Industrial Electronics* 62 (2015), no. 10, pp. 6536–6545

[Ria17]   Riaz, U.: Acceleration of Industrial Analytics Functions on a Platform FPGA. MA thesis. Paderborn University, 2017

[Ste13]   Steinarsson, S.: Downsampling time series for visual representation. PhD thesis. 2013

[Wei]   Weidmüller: *Monitoring Systems GmbH: BLADEcontrol condition monitoring system.* `https://mdcop.weidmueller.com/mediadelivery/asset/900_87890`

[WX06]   Watson, S.; Xiang, J.: Real-time condition monitoring of offshore wind turbines. In: *Proceedings of European Wind Energy Conference & Exhibition (EWEC), Athens, Greece.* Vol. 27. 2006, p. 647654

[WZW⁺13]   Wang, Y.; Zhou, X.; Wang, L.; Yan, J.; Luk, W.; Peng, C.; Tong, J.: SPREAD: A Streaming-Based Partially Reconfigurable Architecture and Programming Model. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21 (12 2013), pp. 2179–2192

Transfer Project T2:
Practical Cryptographic Techniques for Secure and Privacy-Preserving Customer Loyalty Systems

Johannes Blömer[1], Jan Bobolz[2], Fabian Eidens[1], Tibor Jager[3], Paul Kramer[1]

1 Department of Computer Science, Paderborn University, Paderborn, Germany

2 School of Informatics, University of Edinburgh, Scotland, UK [a]

3 School of Electrical, Information and Media Engineering, University of Wuppertal, Wuppertal, Germany

---

[a]Work done while at Paderborn University

1   Introduction

In this transfer project, we designed, implemented, and evaluated a cryptographically secure and privacy-preserving incentive system for retail stores. An incentive system is a system in which customers can participate in promotions to obtain rewards or discounts. They can be deployed to influence the shopping behavior of customers, for example, to achieve a higher customer loyalty rate or increase sales. Further, incentive systems are used as a tool for collecting customer data that serves for instance marketing purposes. Popular incentive systems, e.g., German Payback and American Express Membership Rewards, focus on the latter data-driven business model. Therefore, they are neither privacy-preserving nor do they follow a rule to minimize data collection. They essentially possess a complete record of all members' shopping history in their databases. An infamous example of what can happen if this customer data is not protected properly occurred at the loyalty program of the retailer Target, who in 2012 exposed a girl's pregnancy to her father with coupons sent by Target.[27] Not only individual incidents but also the GDPR as an effort of the EU to protect customer data motivate the need for a privacy-focused alternative.

In this transfer project with Diebold Nixdorf, we designed an incentive system that fills this gap. We visualize the main differences to "classical" incentive systems in Figure 75. The main idea is to replace the large central database that contains the users' shopping records with a token stored on the user's smartphone. Thereby, we drastically reduce the data that can be linked to a user. This token holds the state required for the business logic and is typically a point count computed from the shopping history. We use cryptographic

---

bloemer@upb.de (Johannes Blömer), jan.bobolz@ed.ac.uk (Jan Bobolz), fabian.eidens@uni-paderborn.de (Fabian Eidens), tibor.jager@uni-wuppertal.de (Tibor Jager), paul.kramer@upb.de (Paul Kramer)
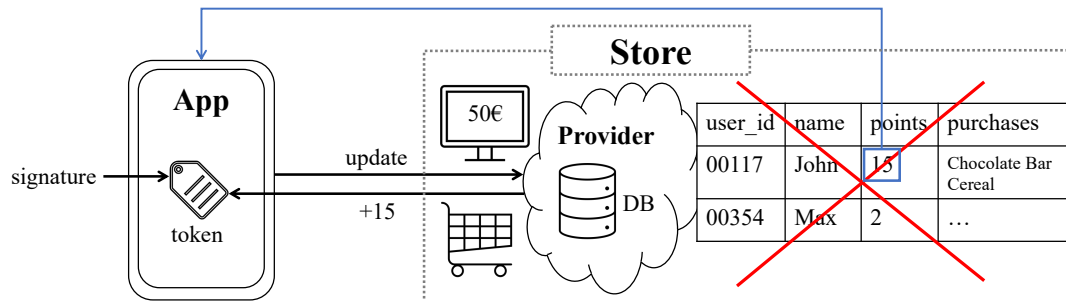[27]https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

Figure 75: *Main differences between our incentive system and "classical" incentive systems. The shopping records with point count in the provider's database are replaced with signed token in app.*

protocols to change the point count of the token when a user wants to collect points for their current basket or trade points for some reward. These protocols, among other properties, ensure privacy and prevent *double-spending attacks*. Double-spending is the canonical attack on token-based systems, in which an attacker copies a digital token and redeems it multiple times.

In addition to designing the incentive system, we implemented a demonstrator and further adjusted the system to address real-world cryptographic challenges. The in-store experience with the demonstrator of the incentive system is as follows: A user goes shopping using their phone and the store's app installed (which includes the incentive system client). In the store, the user scans barcodes of items with their phone, which are then added to a digital basket. In the app, the user can see running promotions that will give them benefits, e.g., buying four chocolate bars and getting one free. The checkout process is also handled in the app and automatically handles the promotion benefits for the user, e.g., checks if the basket contains four chocolate bars and adds a discount for one of them. In our incentive system, these promotions are not limited to a single store visit. Rather, a user can, for example, buy two chocolate bars now and two next time. The incentive system stores this state in a secure token on the user's device.

We started the project with a study of existing cryptographic incentive systems from the literature [MDPD15; JR16; HHNR17] that introduced the concept to let users store their points in an authenticated form. The techniques we developed in Subproject *C1 Robustness and Security* for anonymous credentials and their implementations turned out to be well suited for this, solve open problems, and compensate for some of the downsides of the existing systems. Furthermore, we discussed real-world store infrastructure and details of the currently used hardware with the project partner. We identified the following requirements for a privacy-preserving incentive system.

- Anonymity: Providers are unable to link earn/spend transactions to users. In practice, this protects users from having their shopping history linked to their identity and point values.

- Soundness: Users cannot spend more points than they have earned.

- Online double-spending protection: Given continuous access to a central database, the provider can immediately detect double-spending.

- Offline double-spending protection: Providers can even detect double-spending

in one store without continuous access to a central database. Double-spending transactions can be detected and the perpetrating user can be identified. Losses incurred by double-spending can be reclaimed from that user.

- Partial spending: Users can choose how many points they spend in one transaction.

- Efficiency: The process of earning and spending points can be run on a consumer phone and existing store infrastructure, i.e., the privacy-preserving incentive system is ready for real-world applications.

Our privacy-preserving incentive systems [BBDE19; BEK$^+$20] fulfill all of the goals. Previous systems from the literature do not have offline double-spending protection, they do not support partial spending or, if they support partial spending, the combination with offline double-spending protection is not securely realized. In the following, we present the incentive systems in more detail, how we adapt them in this project, and discuss highlights of the project results. Furthermore, we give important lessons that we learned in the process from theory to practice.

2   Main Contribution

The main privacy issue of current incentive systems in practice is that the incentive system provider stores user data (e.g., each user's point count) in a central database. This architecture practically forces participating users to reveal their identities in order for the correct entry to be updated in the provider's database. Unfortunately, this allows the provider to link purchases to the user's identity, enabling the creation of detailed user profiles. Our main idea to remedy this, as sketched in Figure 75, is to store personal user data (name, current point value) in a cryptographically authenticated token on the user device instead of a central database. We run privacy-preserving protocols that can update the user's data without ever revealing the data or the user's identity to the incentive system provider. For example, we can have a user with k points update their point count by +15. After that procedure, the user will hold a token that certifies $k + 15$ points, while the provider does not learn the old value k, the new value $k + 15$, or any other user data. The provider is only aware that it increments some hidden authenticated value k by +15.

The first iteration of our incentive system [BBDE19] is built from updatable anonymous credentials (UAC), which indeed store authenticated data ("attributes") within an authenticated token ("credential") that can be updated in a privacy-preserving way (see Subproject *C1 Robustness and Security* on page 145). The main idea of an update operation [BBDE19] is that the user first computes the update (e.g., +15) locally, sends this updated data in hidden form (in a cryptographic commitment) to the provider, and proves, with a zero-knowledge proof of knowledge, that the hidden data is a valid update to the user's old authenticated data. The provider then blindly authenticates the updated data.

One of the main challenges when building cryptographic incentive systems is how to handle *double-spending*. Say a user holds a token with 20 points. Of course, the user and provider can run the update protocol for the user to receive an updated token with $20 - 5 = 15$ points. However, we need a mechanism to prevent the user from using the old 20 point token *again* (which would be considered *double-spending*). Note that detecting double-spending is made difficult by our strong privacy aspirations: the act of spending a

token must not reveal any information about the token, hence the token itself is generally hidden. As a consequence, using the same token twice is, by default, not detectable by the provider. One typical way to solve the double-spending issue is *online* double-spending detection. There, every token is assigned a random ID. To invalidate a token (e.g., in the −5 scenario above), the user reveals the token's random ID. The provider checks whether that ID has already been invalidated and, if so, whether the user is trying to double-spend the token. To ensure privacy, the random ID is chosen by the user and only revealed to the provider when invalidating the token. This means that as long as the user does not double-spend, the provider only learns meaningless random numbers through this process. The downside to this approach is that it requires stores to have a consistent connection to a database containing all invalidated IDs. If a store loses the database connection, it cannot check whether a user is double-spending or legitimately spending a valid token. *Offline* double-spending protection (e.g., [HHNR17]) is an additional feature that mitigates this issue. It allows offline stores to speculatively accept a token without checking the ID. If it later turns out that the ID had already been spent (e.g., in another store), the offline double-spending protection feature guarantees that the double-spending user's identity can be revealed, allowing stores to identify misbehaving users and recoup any losses (rewards given erroneously) incurred by undetected double-spending. We combine both sorts of double-spending mitigation in [BBDE19].

Afterward, we identified that in real-world applications, the earning of points is the most frequent operation and should be further optimized. Typically, users have to earn points many times before they can spend them on a reward. In the follow-up paper [BEK+20], we present a new incentive system in which the protocol to earn points works *without* the relatively costly machinery of zero-knowledge proofs of knowledge. This is enabled by using structure-preserving signatures on equivalence classes (SPS-EQ) [FHS19] to authenticate the user data. SPS-EQ come with special randomization features that allow the provider to verify and modify a hidden version of the user's data. Additionally, we add desirable features such as (1) improved privacy for double-spending users (in [BBDE19], double-spending users would incidentally reveal their whole purchase history to the provider) and (2) support for retrying interrupted protocol executions without triggering double-spending protection. We use the ideas from [BEK+20] as the cryptographic basis of our system. However, we have significantly extended the construction and its infrastructure to match the requirements and desirable features of real-world stores.

2.1   Prototype

In the following, we describe how the in-store experience mentioned in the introduction is supported by the incentive system prototype developed in this project. Recall that a user interacts with the system using their smartphone and the app that we developed. At the first start, the app explains key features of the incentive system and then guides the user through a one-time registration process (Figure 76). The result is a join-token in the form of a digital signature on the user identity by the provider. By this, we can later guarantee for the store that the system can identify users to claim any losses, but only if they double-spend. We use the join-token in any interaction with the store in the incentive system to prove that the user is part of the system and therefore a valid user. The app then fetches any running promotions of the store. Promotions are an artifact that the store provider can configure. It
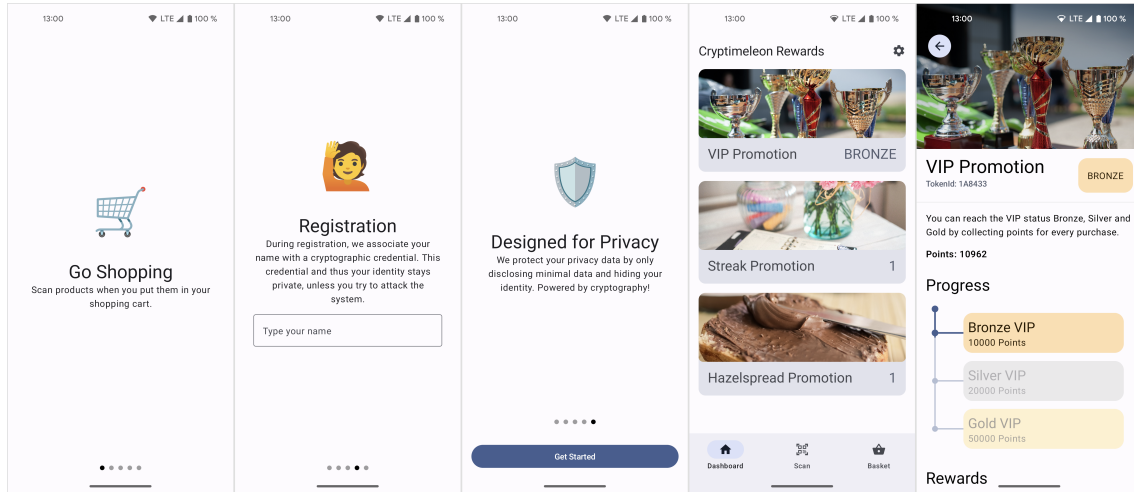
Figure 76: *1.-3. onboarding and registration screens in app. 4) dashboard with an overview of running promotions. 5) detail screen for VIP promotion.*

encapsulates the rules for how users can gain rewards. In the prototype, we support the following promotions with some examples of the rules and rewards:

- discount promotions with the rule "buy x many A" and the reward "get Y for free",

- VIP promotions with the rule "buy x currency worth of products (over multiple visits), become bronze ($x > c_1$), silver ($x > c_2$) or gold VIP ($x > c_3$)" with the reward that the user gets 2%, 5%, or 10% discount on future purchases, and

- streak promotions with the rule "go shopping at least once every 7 days" and the reward "get a free coffee after a streak of 2 weeks".

The user (and app) is now ready for the main part of the shopping, i.e., scanning products and putting them in the digital basket (see Figure 77). Here, the store is not involved and no data is revealed since the app has all the necessary data stored locally. When users are ready for the checkout they can do so directly in the app and leave the store. During the checkout process in the app, the app checks if any promotions can be updated following the rules of the promotions. For example, the user has previously bought 2 chocolate bars (as stored in the user's token), 1 is in the basket, and there is a discount promotion that gives 1 chocolate bar for free if the user has bought at least 3. Then the app interacts with the provider to get a privacy-preserving update on the promotion.

Let us describe the update process in detail. A user has a token for every promotion certifying the status of the promotion. These tokens are obtained by proving ownership of a join-token and initialized with a starting value of 0. A token is a Pedersen commitment [Ped91] on the user's status of the promotion, a cryptographic primitive that hides the data but allows proving statements on the data. In addition, a valid token must be signed with an SPS-EQ signature by the provider. This prevents users from generating valid tokens, i.e., "printing money".

For our example, if the user simply wants to collect 2 points for buying 2 chocolate bars, the app runs the *earn protocol* with the provider: The app sends the randomized token to the provider, who adds 2 points to the token and issues an SPS-EQ signature on the token with +2 points. The app de-randomizes this new token and stores the token worth +2
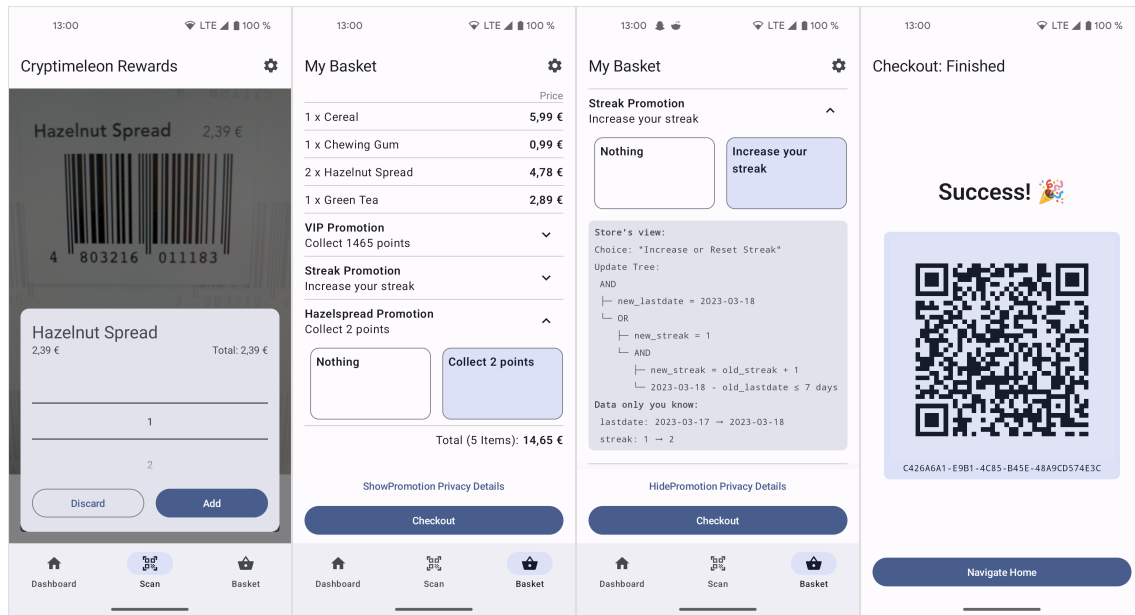
Figure 77: *Walkthrough of the shopping process: 1) Scanning products, 2) selecting updates for promotions, 3) details on privacy consequences of updates, 4) result screen with QR code as proof of payment and for claiming rewards.*

points. This earn protocol only uses the algebraic properties of the SPS-EQ signatures.

At some point, the user will have collected enough points to get some reward, and the app then runs the *spend protocol*. For example, we assume the user collected 5 points, whereas 4 points can be traded in for a free chocolate bar. For this, we utilize non-interactive zero-knowledge arguments of knowledge (NIZKs), namely Schnorr-style Sigma protocols [Sch90]. The app generates a remainder token holding the remaining amount of 1 point. Then, it sends the current token with 5 points and the remainder token with 1 points together with an NIZK that proves the following statements: 1) The old token holds at least 4 points (using a so-called range proof) and 2) The remainder token holds 4 points less than the original token. After verifying the proof, the provider is convinced that the new token has been correctly updated according to the rules of the promotion and signs the remainder token, which becomes the new valid token of the user. Additionally, the reward is added to the user's basket. To prevent double-spending attacks, the old token is invalidated through this procedure, which we explain later in more detail.

All these updates are privacy-preserving, meaning that updates cannot be linked to users, and the exact points counts of the tokens are kept secret. Only necessary data is shared with the provider. We achieve this by the randomization properties of SPS-EQ signatures, the hiding properties of Pedersen commitments, and the zero-knowledge properties of the NIZKs.

So far, we only considered the provider's side as one abstract instance, which corresponds to older versions of the incentive system [BBDE19; BEK+20]. However, it turned out that applying this system to *multiple* stores, e.g., all stores of a supermarket chain, is a non-trivial task: Simply deploying copies of the incentive system at each store is not desirable, because the provider's secret key would be shared among all stores, and hence one compromised store would break the whole system. For availability reasons, we must

keep some functionality in the stores such that users can pay their baskets and update tokens in case of network outages. Our new *multi-store infrastructure* fulfills both requirements (see Figure 78):

- There is exactly one *provider* in the cloud that holds the SPS-EQ secret key. Only the provider can issue SPS-EQ signatures and thus create and update tokens. The SPS-EQ secret key could be stored in a hardware security module (HSM).

- Every *store* has some standard digital signature key pair (ECDSA) that is trusted in the incentive system's public-key infrastructure. The store uses its secret key to authorize the provider to execute a token update.
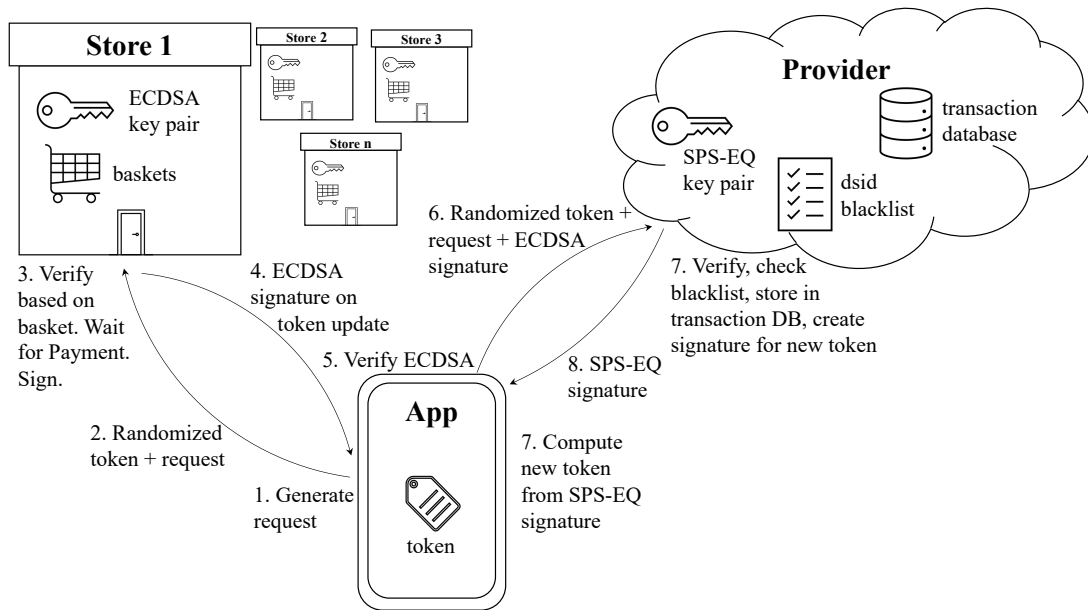

Figure 78: *Multi-store infrastructure.*

First, the user communicates with one of the stores. The store verifies the user's request based on the user's baskets and then issues an ECDSA signature to authorize a token update. For example, if a user wants to earn points for some basket, the store computes and signs the number of points to earn. This process functions in the store even if it is offline, and the user can send the signature to the provider later to obtain an updated token. Then, the provider again performs some verification, checks the store's signature, and then gives some SPS-EQ signature to the user. The user obtained a new token with a valid SPS-EQ signature.

The new multi-store infrastructure enables us to keep the transaction history of users that double-spend secret to the point of double-spending without needing the expensive and complex forward tracing technique from [BEK+20]. Furthermore, during the implementation and discussions with Diebold Nixdorf, we identified the problem of scaling the incentive system to multiple stores. Motivated by this, we introduced the multi-store infrastructure that solved both our scaling problems and enabled *clearing* (i.e., the ability to establish a pool of money that stores pay into when issuing points and can withdraw from when giving out rewards). To summarize, implementing the incentive system made real-world cryptographic challenges visible and lead to improving several aspects of the incentive system.

2.2   Implementation and the Role of Cryptimeleon

The implementation of the incentive system is provided under the open-source MIT license on GitHub.[28] It is powered by the Cryptimeleon project[29] developed at Paderborn University, a collection of cryptography-prototyping libraries written in Java. They provide all necessary primitives, for instance, basic math structures, SPS-EQ signatures, and NIZKs. Further, they natively support the MCL[30] library that ships highly optimized bilinear curves to all relevant architectures. The Android app is implemented in Kotlin with the framework Jetpack Compose. The web services use Java and Spring Boot and are deployed via Docker and Docker Compose; our web app is written in Vue.js.

While there is still room for optimizations and speedups, all implemented protocols run in well under a second and thus are more than practical. We summarize our benchmark results in Table 1.

Protocol	Time (ms)						Size (KB)				
	A^*	$S^\dagger$	A^*	$P^\dagger$	A^*	Total	A→S	S→A	A→P	P→A	Total
Registration	0.0	0.5	0.7	1.9	10.0	13.0	0.4	0.9	1.0	0.8	3.1
Join			23.3	3.2	16.8	43.3			2.1	0.9	2.9
Earn	6.0	0.4	4.1	3.7	14.2	28.4	0.5	0.8	1.9	0.9	4.0
Spend	47.7	21.4	0.5	24.0	15.1	108.7	12.0	1.0	12.3	1.0	26.4

Table 1: *Benchmarks for simple point-collection promotion. Times averaged over* 1000 *runs on Google Pixel 5 *and M1 Macbook Pro* †*. Message sizes were recorded with Wireshark. Abbreviations: app (A), store (S), provider (P).*

3   Impact and Outlook

Right now, privacy is under attack by surveillance capitalism: Corporations collect data about people on a large scale, analyze it, create comprehensive user profiles, and then use those profiles for profit, usually via targeted advertisement. This data collection is already pervasive online. Additionally, digital incentive systems bring data collection to even more areas of everyday life, namely offline shopping.

Even if the provider of a traditional incentive system has good intentions, users have no agency over the data they hand over each time they interact with the incentive system. It is entirely possible, perhaps even likely, that the collected data will eventually be hacked or leaked, or that the provider changes its strategy and starts abusing the collected data.

Privacy-preserving incentive systems, such as the one we have developed, play a crucial role in solving the privacy issues of traditional incentive systems: Users gain cryptographic guarantees that their data *cannot* be collected by the provider. From the point of view of the provider, a privacy-preserving incentive system still offers all the (non-invasive)

---

[28]https://github.com/cryptimeleon/incentive-system
[29]https://cryptimeleon.org
[30]https://github.com/herumi/mcl

benefits for participating stores, giving them a way to reward loyalty, gamify shopping through points, or incentivize the purchase of certain products.

Unfortunately, right now, there seems to be insufficient incentive for stores to deploy a privacy-preserving system instead of one that collects as much data as possible. While users would unequivocally prefer a privacy-preserving solution, in practice, sufficiently many users are ignorant or apathetic towards privacy and are willing to participate in systems without any expectation of privacy. Providers, of course, prefer collecting data over not collecting data. There are two possible ways out of this situation. First, we (as a society) can educate users about the dangers of the unmitigated collection of personal data. If sufficiently many users demand privacy, privacy-preserving systems will gain traction. Second, we can prescribe a privacy-preserving system through law. The GDPR has been incredibly impactful regarding data collection and user consent. In the same vein, mandating the use of privacy-preserving systems could be a highly effective consumer protection law. Projects such as ours take the first important step towards this, proving that such systems can be built with reasonable effort. This is crucial information for lawmakers, who would be understandably hesitant to outlaw systems with no viable alternative.

Projects such as this one are also valuable because they bring together practical aspects and academia. Often, academia only supplies the very first step for building new systems: the very basic ideas. These ideas are motivated and illustrated by idealized scenarios and aim to answer fundamental questions rather than supply a comprehensive blueprint for building concrete systems. As a second step, it is then on the industry to take the academic answers and refine them into a real system. As evidenced by this project, it is sometimes fruitful to involve academia in the second step. This allows the academic side to revisit their idealized assumptions (using insights from the industry) and to improve upon their answers. For example, our new infrastructure with support for multiple stores is a direct result of requirements from our industry partner, improving and even simplifying the system we have built based on idealized assumptions. It is also important to test and benchmark constructions in software for demonstration. Not only does the resulting software serve as a demonstrator, showing that building such systems is realistic, but the process of implementing the system also forces one to consider crucial details previously ignored. Those may even represent future research opportunities or collaboration with industry partners on related topics.

While our system proves that a privacy-preserving approach to incentive systems is viable, there are some open research questions to consider in the future: (1) How we can guard our system against adversaries with access to a hypothetical large quantum computer? (2) What are further applications for the techniques we used for incentive systems (e.g., electronic cash)? (3) How can we make the public more cognizant of the privacy issues of currently deployed systems, and of the alternatives enabled by modern cryptography?

Bibliography

[BBDE19]   BLÖMER, J.; BOBOLZ, J.; DIEMERT, D.; EIDENS, F.: Updatable Anonymous Credentials and Applications to Incentive Systems. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. ACM Press, 2019, pp. 1671–1685

[BEK+20]   Bobolz, J.; Eidens, F.; Krenn, S.; Slamanig, D.; Striecks, C.: Privacy-Preserving Incentive
           Systems with Highly Efficient Point-Collection. In: *ASIACCS 20: 15th ACM Symposium
           on Information, Computer and Communications Security*. Ed. by Sun, H.-M.; Shieh, S.-P.;
           Gu, G.; Ateniese, G. ACM Press, Oct. 2020, pp. 319–333

[FHS19]   Fuchsbauer, G.; Hanser, C.; Slamanig, D.: Structure-Preserving Signatures on Equivalence
           Classes and Constant-Size Anonymous Credentials. In: *Journal of Cryptology* 32 (Apr.
           2019), no. 2, pp. 498–546

[HHNR17]   Hartung, G.; Hoffmann, M.; Nagel, M.; Rupp, A.: BBA+: Improving the Security and
           Applicability of Privacy-Preserving Point Collection. In: *ACM CCS 2017: 24th Conference
           on Computer and Communications Security*. Ed. by Thuraisingham, B. M.; Evans, D.;
           Malkin, T.; Xu, D. ACM Press, Oct. 2017, pp. 1925–1942

[JR16]   Jager, T.; Rupp, A.: Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving
           Way. In: *Proceedings on Privacy Enhancing Technologies* 2016 (July 2016), no. 3, pp. 62–
           82

[MDPD15]   Milutinovic, M.; Dacosta, I.; Put, A.; Decker, B. D.: uCentive: An Efficient, Anony-
           mous and Unlinkable Incentives Scheme. In: *TrustCom/BigDataSE/ISPA (1)*. IEEE, 2015,
           pp. 588–595.

[Ped91]   Pedersen, T. P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Shar-
           ing. In: *Proceedings of the 11th Annual International Cryptology Conference on Advances
           in Cryptology*. CRYPTO '91. Berlin, Heidelberg: Springer-Verlag, 1991, pp. 129–140

[Sch90]   Schnorr, C. P.: Efficient Identification and Signatures for Smart Cards. In: *Advances in
           Cryptology — CRYPTO' 89 Proceedings*. Ed. by Brassard, G. New York, NY: Springer
           New York, 1990, pp. 239–252

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 386 SCHNEIDER, M.: Spezifikationstechnik zur Beschreibung und Analyse von Wertschöpfungssystemen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 386, Paderborn, 2018 – ISBN 978-3-947647-05-7

Bd. 387 ECHTERHOFF, B.: Methodik zur Einführung innovativer Geschäftsmodelle in etablierten Unternehmen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 387, Paderborn, 2018 – ISBN 978-3-947647-06-4

Bd. 388 KRUSE, D.: Teilautomatisierte Parameteridentifikation für die Validierung von Dynamikmodellen im modellbasierten Entwurf mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 388, Paderborn, 2019 – ISBN 978-3-947647-07-1

Bd. 389 MITTAG, T.: Systematik zur Gestaltung der Wertschöpfung für digitalisierte hybride Marktleistungen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 389, Paderborn, 2019 – ISBN 978-3-947647-08-8

Bd. 390 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 15. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 21. und 22. November 2019, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 390, Paderborn, 2019 – ISBN 978-3-947647-09-5

Bd. 391 SCHIERBAUM, A.: Systematik zur Ableitung bedarfsgerechter Systems Engineering Leitfäden im Maschinenbau. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 391, Paderborn, 2019 – ISBN 978-3-947647-10-1

Bd. 392 PAI, A.: Computationally Efficient Modelling and Precision Position and Force Control of SMA Actuators. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 392, Paderborn, 2019 – ISBN 978-3-947647-11-8

Bd. 393 ECHTERFELD, J.: Systematik zur Digitalisierung von Produktprogrammen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 393, Paderborn, 2020 – ISBN 978-3-947647-12-5

Bd. 394 LOCHBICHLER, M.: Systematische Wahl einer Modellierungstiefe im Entwurfsprozess mechatronischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 394, Paderborn, 2020 – ISBN 978-3-947647-13-2

Bd. 395 LUKEI, M.: Systematik zur integrativen Entwicklung von mechatronischen Produkten und deren Prüfmittel. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 395, Paderborn, 2020 – ISBN 978-3-947647-14-9

Bd. 396 KOHLSTEDT, A.: Modellbasierte Synthese einer hybriden Kraft-/Positionsregelung für einen Fahrzeugachsprüfstand mit hydraulischem Hexapod. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 396, Paderborn, 2021 – ISBN 978-3-947647-15-6

Bd. 397 DREWEL, M.: Systematik zum Einstieg in die Plattformökonomie. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 397, Paderborn, 2021 – ISBN 978-3-947647-16-3

Bd. 398 FRANK, M.: Systematik zur Planung des organisationalen Wandels zum Smart Service-Anbieter. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 398, Paderborn, 2021 – ISBN 978-3-947647-17-0

Bd. 399 KOLDEWEY, C.: Systematik zur Entwicklung von Smart Service-Strategien im produzierenden Gewerbe. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 399, Paderborn, 2021 – ISBN 978-3-947647-18-7

Zuletzt erschienene Bände der Verlagsschriftenreihe des Heinz Nixdorf Instituts

Bd. 400 GAUSEMEIER, J. (Hrsg.): Vorausschau und Technologieplanung. 16. Symposium für Vorausschau und Technologieplanung, Heinz Nixdorf Institut, 2. und 3. Dezember 2021, Berlin-Brandenburgische Akademie der Wissenschaften, Berlin, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 400, Paderborn, 2021 – ISBN 978-3-947647-19-4

Bd. 401 BRETZ, L.: Rahmenwerk zur Planung und Einführung von Systems Engineering und Model-Based Systems Engineering. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 401, Paderborn, 2021 – ISBN 978-3-947647-20-0

Bd. 402 WU, L.: Ultrabreitbandige Sampler in SiGe-BiCMOS-Technologie für Analog-Digital-Wandler mit zeitversetzter Abtastung. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 402, Paderborn, 2021 – ISBN 978-3-947647-21-7

Bd. 403 HILLEBRAND, M.: Entwicklungssystematik zur Integration von Eigenschaften der Selbstheilung in Intelligente Technische Systeme. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 403, Paderborn, 2021 – ISBN 978-3-947647-22-4

Bd. 404 OLMA, S.: Systemtheorie von Hardware-in-the-Loop-Simulationen mit Anwendung auf einem Fahrzeugachsprüfstand mit parallelkinematischem Lastsimulator. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 404, Paderborn, 2022 – ISBN 978-3-947647-23-1

Bd. 405 FECHTELPETER, C.: Rahmenwerk zur Gestaltung des Technologietransfers in mittelständisch geprägten Innovationsclustern. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 405, Paderborn, 2022 – ISBN 978-3-947647-24-8

Bd. 406 OLEFF, C.: Proaktives Management von Anforderungsänderungen in der Entwicklung komplexer technischer Systeme. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 406, Paderborn, 2022 – ISBN 978-3-947647-25-5

Bd. 407 JAVED, A. R.: Mixed-Signal Baseband Circuit Design for High Data Rate Wireless Communication in Bulk CMOS and SiGe BiCMOS Technologies. Dissertation, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 407, Paderborn, 2022 – ISBN 978-3-947647-26-2

Bd. 408 DUMITRESCU, R, KOLDEWEY, C.: Datengestützte Projektplanung. Fachbuch. Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 408, Paderborn, 2022 – ISBN 978-3-947647-27-9

Bd. 409 PÖHLER, A.: Automatisierte dezentrale Produktionssteuerung für cyber-physische Produktionssysteme mit digitaler Repräsentation der Beschäftigten. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 409, Paderborn, 2022 – ISBN 978-3-947647-28-6

Bd. 410 RÜDDENKLAU, N.: Hardware-in-the-Loop-Simulation von HD-Scheinwerfer-Steuergeräten zur Entwicklung von Lichtfunktionen in virtuellen Nachtfahrten. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 410, Paderborn, 2023 – ISBN 978-3-947647-29-3

Bd. 411 BIEMELT, P.: Entwurf und Analyse modellprädiktiver Regelungsansätze zur Steigerung des Immersionsempfindens in interaktiven Fahrsimulationen. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Band 411, Paderborn, 2023 – ISBN 978-3-947647-30-9

Das Heinz Nixdorf Institut –
Interdisziplinäres Forschungszentrum
für Informatik und Technik

Das Heinz Nixdorf Institut ist ein Forschungszentrum der Universität Paderborn. Es entstand 1987 aus der Initiative und mit Förderung von Heinz Nixdorf. Damit wollte er Ingenieurwissenschaften und Informatik zusammenführen, um wesentliche Impulse für neue Produkte und Dienstleistungen zu erzeugen. Dies schließt auch die Wechselwirkungen mit dem gesellschaftlichen Umfeld ein.

Die Forschungsarbeit orientiert sich an dem Programm „Dynamik, Vernetzung, Autonomie: Neue Methoden und Technologien für die intelligenten technischen Systeme von morgen". In der Lehre engagiert sich das Heinz Nixdorf Institut in Studiengängen der Informatik, der Ingenieurwissenschaften und der Wirtschaftswissenschaften.

Heute wirken am Heinz Nixdorf Institut acht Professoren/in mit insgesamt 120 Mitarbeiterinnen und Mitarbeitern. Pro Jahr promovieren hier etwa 15 Nachwuchswissenschaftlerinnen und Nachwuchswissenschaftler.

Heinz Nixdorf Institute –
Interdisciplinary Research Centre
for Computer Science and Technology

The Heinz Nixdorf Institute is a research centre within the University of Paderborn. It was founded in 1987 initiated and supported by Heinz Nixdorf. By doing so he wanted to create a symbiosis of computer science and engineering in order to provide critical impetus for new products and services. This includes interactions with the social environment.

Our research is aligned with the program "Dynamics, Networking, Autonomy: New methods and technologies for intelligent technical systems of tomorrow". In training and education the Heinz Nixdorf Institute is involved in many programs of study at the University of Paderborn. The superior goal in education and training is to communicate competencies that are critical in tomorrows economy.

Today eight Professors and 120 researchers work at the Heinz Nixdorf Institute. Per year approximately 15 young researchers receive a doctorate.