

# **Scaffolding Decentralized Community Information Systems for Lifelong Learning Communities**

Von der Fakultät für Mathematik, Informatik und  
Naturwissenschaften der RWTH Aachen University zur Erlangung  
des akademischen Grades eines Doktors der Naturwissenschaften  
genehmigte Dissertation

vorgelegt von

**Peter Marcel de Lange, M.Sc.**

aus Eschweiler, Deutschland

Berichter: Univ.-Prof. Dr. rer. pol. Matthias Jarke  
Priv.-Doz. Dr. rer. nat. Ralf Klamma  
Univ.-Prof. Dr. phil. Martina Ziefle

Tag der mündlichen Prüfung: 20.07.2021

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.



## Abstract

Initially, the Web was developed as a decentralized system of information repositories that facilitate organizational knowledge transfer by allowing anyone to create and access content. However, Web publishing required both technical expertise and hardware infrastructure. With the rise of the Web 2.0, social networking sites and content management systems enabled all users to create Web content. But it simultaneously put the users at the mercy of the platform operators. Services could be shut down, erasing content and disrupting communities.

Decentralized community information systems radically change this dynamic by establishing participants as equal peers, which form a self-governing community. This way, a community regains control over their data, while being able to scale the infrastructure according to their needs.

In this dissertation, we followed a design science approach that provides support for communities to create and host their own decentralized community information systems. On the one hand, we produced several artifacts to provide possible answers to the question of what properties such an infrastructure needs to fulfill. With the *blockchain-based decentralized service registry*, we propose a solution for making community knowledge accessible in a secure and verifiable way. On the other hand, we transfer the metaphor of educational scaffolding to the domain of service development. It is based on the idea, that a scaffold serves as a temporary supporting structure during a building's construction phase. As the construction site develops and the building gets completed, the scaffold gradually gets removed up to the point, that it is not needed anymore. With the *community application editor*, communities are provided with such a scaffolding environment for requirements elicitation, wireframing, modeling and coding their decentralized community applications. Once deployed on the infrastructure, those applications and development efforts remain available, even after the contributing members might have left, serving as the community's long term memory.

We demonstrated and evaluated our artifacts on a European scale, with three longitudinal studies conducted within several communities from different areas of technology enhanced learning, such as the European voluntary service, vocational and educational training providers and in higher education mentoring scenarios. All in all, this shift from data being stored in centralized repositories to a decentralized infrastructure, hosted by community members, opens up possibilities for a more democratic and egalitarian management of community knowledge.





## Kurzfassung

Das Web wurde ursprünglich als dezentrales System von Informationen entwickelt, welches organisationalen Wissensaustausch durch alle ermöglichen sollte. Dieser Austausch erforderte jedoch technische Expertise und Hardware Infrastruktur. Mit dem Schritt zum Web 2.0 wurde das Erstellen und Teilen von Inhalten durch das Aufkommen sozialer Netzwerke und Content Management Systemen allen Nutzern ermöglicht. Hierdurch wurde jedoch gleichzeitig eine Abhängigkeit zu den Plattformanbietern eingegangen. Dienste konnten jederzeit eingestellt werden, was zum Verschwinden von Inhalten führen und schlussendlich Gemeinschaften auseinander bringen kann. Dezentrale Community-Informationssysteme verändern diese Dynamik, indem sie untereinander gleichberechtigte Teilnehmer einer selbstverwalteten Gemeinschaft etablieren. Auf diese Weise erhält eine Gemeinschaft die Kontrolle über ihre Daten zurück und kann die Infrastruktur entsprechend den eigenen Anforderungen skalieren.

Diese Dissertation verfolgt einen Design Science Ansatz, der Gemeinschaften dabei unterstützt, ihre eigenen dezentralen Community-Informationssysteme zu Erstellen und zu Hosten. Zum einen haben wir mehrere Artefakte erstellt, um die Eigenschaften einer solchen Infrastruktur zu erfüllen. Mit der *Blockchain-basierten Service Registry* schlagen wir eine Lösung vor, die Wissen auf sichere und überprüfbare Weise zugänglich macht. Zum anderen übertragen wir die Metapher des Educational Scaffolding (dt.: Bildungs-Gerüstbau) auf den Bereich der Dienst-Entwicklung. Dieses basiert auf der Idee, dass ein Gerüst während der Bauphase eines Gebäudes als temporär unterstützende Struktur dient. Mit dem *Community Application Editor* erhalten Gemeinschaften eine solche Scaffolding Umgebung für das Ermitteln von Anforderungen, das Wireframing, die Modellierung und das Programmieren ihrer dezentralen Anwendungen. Einmal in der Infrastruktur bereitgestellt, bleiben diese Anwendungen auch nach dem Verlassen der beitragenden Mitglieder verfügbar und dienen als Langzeitgedächtnis der Gemeinschaft.

Wir haben unsere Artefakte im Europäischen Kontext demonstriert und evaluiert. Drei Längsschnittstudien wurden in mehreren Gemeinschaften aus verschiedenen Bereichen des technologiegestützten Lernens durchgeführt, unter anderem dem europäischen Freiwilligendienst, mit Anbietern und Nutzern von Weiterbildungsmaßnahmen, sowie in Mentoring-Szenarien für die Hochschulbildung. Zusammenfassend öffnet diese Verlagerung von Daten in eine selbst-gehostete, dezentrale Infrastruktur Möglichkeiten für ein demokratischeres und egalitäreres Management von Wissen.



## Acknowledgments

With this (too) short text, I would like to express my deepest gratitude to all who actively helped and contributed to pushing my work forward, who cheered me up when I didn't see any value in moving forward, who adjusted my path when I was moving into the wrong direction, or who were just there for me to walk a step along this way together. One really cannot do this alone.

I would like to start by providing my gratitude to my dissertation committee. My appreciation goes out to Prof Dr. Matthias Jarke, for granting me the opportunity to start (and finish) my research at i5. I still recall many wise words and thoughts from our conversations that shaped the way I see some things – professionally and scientifically – now. I want to thank PD Dr. Ralf Klamma for providing me a place to work. Starting as a working student back in 2012, I spend my whole academic professional life in his group, and it surely shaped me. From time to time, I know I can be stubborn and react strongly to criticism, and Ralf always found a way to get through to me and put me back on track. I am deeply grateful for this. Furthermore, I would like to thank Prof. Dr. Martina Ziefle for providing an additional perspective on my work from an external position. Finally, I thank PD Dr. Walter Unger for accepting my request to act as examiner and Prof. Dr. Matthias Müller for being a very relaxed and ensuring head of exam.

My appreciation goes out to my colleagues at ACIS, who created a very unique working atmosphere, which I enjoyed very much. Following the order appearance to me, I would like to start by thanking Dr. Dominik Renzel, who provided me the student worker opportunity that started my life at i5. I am deeply grateful to Dr. Petru Nicolaescu, who not only supervised my Master thesis before becoming a colleague and mentor for my first three years, but who also became a great friend during and beyond this time. I'd like to thank Dr. István Koren (which I'd never forget mentioning here) for challenging me professionally from time to time during my first years, and for providing invaluable practical help and tips throughout the final stages of my thesis work. Furthermore, I would like to thank Dr. Mohsen Shahriari, Dr. Milos Kravcik, Georgios Toubekis, Katja Neulinger, and Michal Słupczyński. Anyone of you has provided me with help in some way throughout these years and I am truly grateful for that.

My gratitude goes out to Tatjana Liberzon and Reinhard Linde, for their assistance on technical matters. Especially Reinhard was often challenged by me with short term requests that he always fulfilled to the fullest. I thank Daniele Glöckner, Claudia Puhl, Leany Maassen and Romina Reddig for always providing exceptional administrative services.

Furthermore, I want to thank Alexander Neumann, Lars Gleim and Sascha Welten. I thank Alex for providing me support on many technical matters, and generally for being a great help throughout my final years at the chair. I thank Lars for being such a great colleague when we “took over” the organization of the database lecture exercise class, and I thank Sascha for later taking over these responsibilities from us. Of this group, I shared

the longest walk with Alex, who started his student worker career as my first “HiWi” back in 2016, and continued to be first my Master thesis student, then colleague, and in some way successor in the ACIS group. But – and really this is most important – I want to thank each of you equally for being part of this unholy alliance of four, which shaped my working (and some of my private) life during my final two years at the chair.

My gratefulness also goes out (again in the order of appearance to me) to Kim Fidomski, Felix Schwinger, Stefan Braun, Johannes Lipp and Yongli Mou for being such great colleagues. I have fond memories of our evenings at the chair and in Aachen’s Kiste.

As well, I want to thank Prof. Dr. Stefan Decker for providing me the opportunity to continue my work at the chair after he took over. Being able to finish the work started under a different head of institute is by no means guaranteed and Prof. Decker did an excellent job in this transitioning phase for all of us doctoral thesis workers.

My acknowledgment section would not be complete without mentioning two external colleagues that shaped this work so much. My highest appreciation goes out to (soon to be) Dr. Bernhard Göschlberger and Dr. Tracie Farrell, who provided so much input to this thesis and to my personal development. I am grateful for Bernhard’s work on the technical parts of our project, and I am grateful for Tracie’s exceptional evaluation- and writing skills of course, but I am even more grateful for the personal support they provided me on countless occasions. Another unholy alliance that was, really.

I would like to thank all those students that decided to do their thesis with me. In order of appearance, I am grateful for the contributions of Thomas Winkler [Wink16], Jan Benscheid [Bens16], Adam Brunmeier [Brun17], Mario Rosenstengel [Rose17], Melisa Cecilia [Ceci18], Alexander Neumann [Neum18], Tom Janson [Jans19], Philipp Hossner [Hoss19], Philipp Roytburg [Royt19], Niels Wiessner [Wies19], Michal Słupczyński [Slup20], Philipp Dolif [Doli20], Aaron Conrardy [Conr20], Julius Rickert [Rick20], Lennart Bengtson [Beng20] and Erdzan Rastoder [Rast21]. Moreover, my appreciation goes out for the work done by all the student workers that worked under my supervision. In alphabetical order, I would like to thank Enes Aldemir, Frederik Basels, Jan Benscheid, Thomas Cujé, Aaron Conrardy, Navid Rahimi Danesh, Philipp Dolif, Boris Jovanovic, Jonas Könnig, Michael Kretschmer, Leonardo da Matta, Mallika Mewar, Jasper Nalbach, Alexander Neumann, Erdzan Rastodor, Klea Sanka, as well as all those students who supported me in my teaching activities.

Last, but by no means least, I want to thank my family and friends. I have been blessed with a wonderful wife, the best family one can wish for, and friends with whom I can not only enjoy the good times, but on whom I can always count on when times go hard. I value this dearly.

So, with that being said..it has been an interesting ride. Never boring, often fun, always educational. Off to new horizons then, Reinbügeln and Cheers!

Munich, July 2021  
Peter de Lange

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                    | <b>1</b>  |
| 1.1      | Motivating Use Case . . . . .                          | 2         |
| 1.2      | Solution Concepts and Research Questions . . . . .     | 3         |
| 1.3      | Thesis Contribution and Research Context . . . . .     | 5         |
| 1.4      | Thesis Outline . . . . .                               | 8         |
| <b>2</b> | <b>Research Context</b>                                | <b>9</b>  |
| 2.1      | Decentralized Architectures . . . . .                  | 9         |
| 2.1.1    | P2P Networks . . . . .                                 | 9         |
| 2.1.2    | Microservices . . . . .                                | 11        |
| 2.2      | Microservice Registry and Discovery . . . . .          | 12        |
| 2.3      | Distributed Consensus . . . . .                        | 15        |
| 2.3.1    | Consensus Mechanisms and the Blockchain . . . . .      | 16        |
| 2.4      | Incentivation and Reputation . . . . .                 | 19        |
| 2.4.1    | Trust . . . . .  | 21        |
| 2.4.2    | Reputation Systems . . . . .                           | 21        |
| 2.5      | Scaffolding . . . . .                                  | 24        |
| 2.5.1    | Usage in Digital Learning Environments . . . . .       | 24        |
| 2.5.2    | Scaffolding of Community Information Systems . . . . . | 26        |
| 2.6      | Social Bots . . . . .                                  | 29        |
| 2.7      | Learning in CoPs . . . . .                             | 30        |
| 2.7.1    | Monitoring of Learning Analytics Data . . . . .        | 31        |
| 2.7.2    | Knowledge Building . . . . .                           | 32        |
| <b>3</b> | <b>Decentralized CIS Infrastructures</b>               | <b>35</b> |
| 3.1      | Methodology . . . . .                                  | 36        |
| 3.2      | Concept and Technical Foundation . . . . .             | 37        |
| 3.3      | Microservice Discovery with API Metadata . . . . .     | 40        |

|          |  |           |
|----------|--|-----------|
| 3.3.1    | Motivation . . . . .                                     | 41        |
| 3.3.2    | Concept . . . . .  | 41        |
| 3.3.3    | Realization . . . . .                                    | 44        |
| 3.3.4    | Evaluation . . . . .                                     | 46        |
| 3.4      | Decentralized Service Registry . . . . .                 | 48        |
| 3.4.1    | Motivation . . . . .                                     | 48        |
| 3.4.2    | Concept . . . . .  | 51        |
| 3.4.3    | Realization . . . . .                                    | 53        |
| 3.4.4    | Evaluation . . . . .                                     | 56        |
| 3.5      | Community Contribution Incentivization . . . . .         | 59        |
| 3.5.1    | Motivation . . . . .                                     | 59        |
| 3.5.2    | Concept . . . . .  | 60        |
| 3.5.3    | Realization . . . . .                                    | 62        |
| 3.5.4    | Evaluation . . . . .                                     | 67        |
| 3.6      | Verification and Consent Management of LA Data . . . . . | 70        |
| 3.6.1    | Motivation . . . . .                                     | 70        |
| 3.6.2    | Concept . . . . .  | 72        |
| 3.6.3    | Realization . . . . .                                    | 74        |
| 3.6.4    | Evaluation . . . . .                                     | 76        |
| 3.7      | Discussion and Conclusion . . . . .                      | 80        |
| <b>4</b> | <b>Scaffolding Decentralized CIS</b>                     | <b>83</b> |
| 4.1      | Methodology . . . . .                                    | 84        |
| 4.2      | Motivation for Model-Driven Scaffolding . . . . .        | 86        |
| 4.3      | Conceptual Overview . . . . .                            | 88        |
| 4.3.1    | View-based MDWE . . . . .                                | 88        |
| 4.3.2    | Web Application Metamodel . . . . .                      | 90        |
| 4.3.3    | Role-based Project Management . . . . .                  | 92        |
| 4.3.4    | A NRT Evaluation Center . . . . .                        | 94        |
| 4.3.5    | Model Synchronization for Live Code Editing . . . . .    | 96        |
| 4.3.6    | Integration of Wireframing Support for MDWE . . . . .    | 97        |
| 4.4      | Realization . . . . .                                    | 100       |
| 4.4.1    | User Interface . . . . .                                 | 100       |
| 4.4.2    | Versioning System . . . . .                              | 103       |
| 4.4.3    | Architectural Overview . . . . .                         | 104       |
| 4.4.4    | Wireframe Model Transformations . . . . .                | 106       |
| 4.5      | Evaluation . . . . .                                     | 108       |
| 4.5.1    | Initial Evaluation . . . . .                             | 108       |

|          |  |            |
|----------|--|------------|
| 4.5.2    | Evaluation with Heterogeneous Teams . . . . .                  | 109        |
| 4.5.3    | Evaluation in a Lab Course . . . . .                           | 110        |
| 4.5.4    | Live Code Editor Evaluation . . . . .                          | 111        |
| 4.5.5    | Wireframing User Evaluation . . . . .                          | 112        |
| 4.5.6    | Wireframing Activity Evaluation . . . . .                      | 113        |
| 4.5.7    | Service Success Measurement Evaluation . . . . .               | 114        |
| 4.5.8    | Project Management Evaluation . . . . .                        | 115        |
| 4.6      | SBF: The Social Bot Framework . . . . .                        | 117        |
| 4.6.1    | Motivation . . . . .   | 118        |
| 4.6.2    | Concept: A Metamodel for Social Bots . . . . .                 | 118        |
| 4.6.3    | Realization: Social Bot Life-Cycle . . . . .                   | 120        |
| 4.6.4    | Evaluation . . . . .   | 124        |
| 4.7      | Discussion and Conclusion . . . . .                            | 127        |
| <b>5</b> | <b>Distributed Learning in Decentralized CIS</b>               | <b>129</b> |
| 5.1      | A Virtual Vocational Training Center . . . . .                 | 131        |
| 5.1.1    | Introduction . . . . .   | 131        |
| 5.1.2    | Use Case: Educational Vocational Training . . . . .            | 133        |
| 5.1.3    | Realization . . . . .  | 134        |
| 5.1.4    | Evaluation . . . . .   | 138        |
| 5.1.5    | Summary . . . . .  | 147        |
| 5.2      | Infrastructure for Knowledge Building . . . . .                | 147        |
| 5.2.1    | Introduction . . . . .   | 147        |
| 5.2.2    | Methodology . . . . .  | 148        |
| 5.2.3    | Digital Question-Based Dialog For Ignorance Modeling . . . . . | 150        |
| 5.2.4    | Realizing the Distributed Oracle . . . . .                     | 151        |
| 5.2.5    | Evaluation . . . . .   | 156        |
| 5.2.6    | Summary . . . . .  | 171        |
| 5.3      | A Distributed Mentoring Architecture . . . . .                 | 172        |
| 5.3.1    | Introduction and Motivation . . . . .                          | 172        |
| 5.3.2    | Architecture . . . . .   | 173        |
| 5.3.3    | Summary and Outlook . . . . .                                  | 175        |
| 5.4      | Discussion and Conclusion . . . . .                            | 175        |
| <b>6</b> | <b>Conclusion and Future Work</b>                              | <b>177</b> |
| 6.1      | Conclusion . . . . .   | 177        |
| 6.2      | Future Work and Outlook . . . . .                              | 180        |

|                        |                                     |
|------------------------|-------------------------------------|
| <b>References</b>      | <b>183</b>                          |
| <b>List of Figures</b> | <b>209</b>                          |
| <b>List of Tables</b>  | <b>213</b>                          |
| <b>Appendices</b>      | <b>215</b>                          |
| Appendix A             | List of Abbreviations . . . . . 215 |
| Appendix B             | Own Publications . . . . . 223      |
| Appendix C             | Curriculum Vitae . . . . . 229      |



# Chapter 1

## Introduction

When Tim Berners-Lee proposed the Web in 1989, he envisioned a decentralized system of information repositories that facilitate organizational knowledge transfer by allowing anyone to create, reference, and access content [Bern89]. However, Web authoring and publication required both technical expertise and hardware infrastructure. With the rise of the Web 2.0 in the early 2000s, Social Networking Sites (SNS) and Content Management Systems (CMS) enabled all users to create Web content [ORei07]. But it simultaneously put the users at the mercy of the platform operators. Services could suddenly be shut down, erasing content and disrupting communities. As well, private data is often stored insecurely, used for commercial purposes, or even revealed in data breaches. The proprietary nature of the vast majority of these platforms leaves users little bargaining power to change those terms.

Decentralized Community Information System (CIS) radically change this dynamic by establishing participants as equal peers, which form a self-governing community. A Peer to Peer (P2P) structure can provide scalability and distribute the utilization of computing resources. In combination with public key cryptography, it allows users to sign messages and store private data securely, providing privacy without relying on trusted infrastructure. It is clear that these properties are especially appealing to online Communities of Practice (CoPs) [Weng98]. These communities are not bound together by an organization, but rather by sharing a common craft or profession, with the desire to learn from each other through knowledge sharing and knowledge building.

In practice though, only few CoPs have the size and influence to get tools tailored to their needs. The long tail [Ande06] of CoPs does not possess the resources, such as hosting infrastructures or shared budget. In their working

practices, most CoPs adopt publicly available tools (e.g. social software) and repurpose them according to their needs, mitigating the tools' technical shortcomings through socially enforced usage policies. These (mostly unwritten) policies include the knowledge necessary to navigate within the digital community space and are an entry barrier for novices as well as a hindrance to community coherence. Moreover, the CoP becomes dependent on the tool provider and also loses control over its data. Even if a CoP manages to establish a centralized infrastructure, this often results in dependencies on single, knowledgeable members or institutions and does not account for dynamic membership, a common characteristic of CoPs.

## 1.1 Motivating Use Case

As an introductory example, we consider a community of young European youth workers, which are preparing for participation in a European-funded training course on “creative leadership”. The participants are an international group, with different levels of experience, from multiple organizations and countries. While they may not yet constitute a CoP, these young adults form a Community of Inquiry (CoI) as a precursor to identifying areas of shared practice [LYPe14], eventually leading to a CoP. The trainer team must create learning content that appeals to this diverse group and meets their needs, which is a challenge, given the complexity of both creativity and leadership as learning subjects. In addition, the three trainers providing the course are distributed across different countries and organizations as well, with no possibility to meet beforehand. Since the whole CoP neither shares a geographic location, nor central infrastructure or budget, this use case stands exemplary for the needs and challenges of distributed CoPs. To help establish the boundaries of the participants' knowledge and identify common ground or potential conflicts, the trainers want to find out which questions the participants have about creative leadership and how those questions relate to one another. In its analog form, this involves an on-scene session at the start of the training course, where the community has a limited time-frame to establish their community ignorance by writing down questions they have. A digital version of the concept, hosted decentrally by the community itself, could be applied already before the community meets.

As a consequence, we claim that a suitable infrastructure for CoPs needs to be decentralized and managed by the community members themselves. It should be easily deployable, extensible and flexible in terms of scalability and accessibility from the outside. The microservice paradigm with loosely coupled services,

bound together by lightweight protocols, fits these demands perfectly. Combined with an underlying P2P network of nodes managed by the CoPs themselves, the microservices should self-replicate through the network according to the community's current needs.

A scaffold serves as a temporary supporting structure during a building's construction phase. As the construction site develops and the building gets completed, the scaffold gradually gets removed up to the point, that it is not needed anymore. The metaphor of using *educational scaffolding*, a term first coined in the 70s by Wood et al. [WBRo76], describes the process of teachers assisting students in their learning process. By applying the principles of scaffolding from the educational domain to the domain of service development for CoPs, we provide communities with the means to develop these microservices collaboratively, integrating all members into the requirements analysis, design and success evaluation. Once deployed on the infrastructure, those services and development efforts should remain available, even after the contributing member has left the CoP. Like the ship in the Theseus paradox, a community should be able to persist, even though all of its members have changed over time, as long as there are people willing to engage. Serving as a community's long term memory, the infrastructure allows members to learn from their "ancestors", much like we can observe in scientific communities. Just like opening the water tap, using a certain learning environment should be available to every community member at all times. Thus, we propose a *Learning as a Utility* approach, which makes it possible for all community members to equally engage in development, hosting and using learning applications.

## 1.2 Solution Concepts and Research Questions

In this dissertation, we propose a decentralized approach for designing and operating CIS. Based on modern Web development standards, we provide two areas of contribution to the domain of CIS research. The first is a methodology and scaffolding support to collaboratively create decentralized applications, as well as the infrastructure to run it on. Our development methodology combines the strengths of proven Model-Driven Web Engineering (MDWE) techniques with requirements collection and analysis support, as well as service success evaluation. We add to this decentralized application development cycle the possibilities of collaborative wireframing to further integrate all CoP members into the development process. The role of social bots as a means to interact with an application is also investigated.

Our second contribution is the decentralized CIS infrastructure, based on the P2P paradigm. It is supported by blockchain technology for immutable and tamper-proof access to the CoP's collective knowledge. Here, we investigate the role of a decentralized service registry and discovery, and propose means to incentivize community contributions, based on blockchain currency reimbursements.

Our approach has been evaluated within three large demonstrations in the domain of Technology Enhanced Learning (TEL), and these evaluations also built the basis for many requirements and research questions continuously answered during the course of this dissertation. These evaluations, next to the contributions to scaffolding and operating decentralized CIS, build the third pillar of this dissertation and have been performed in the domains of educational vocational training, the European voluntary service and within university mentoring processes. With our contributions, we aim at answering the following research questions:

**RQ 1 - Decentralized Infrastructures: What properties does a decentralized, self-hosted infrastructure for CIS need to fulfill?** As mentioned, current tool support for CoPs usually is based on centralized infrastructures. While decentralization and self-hosting bears potential for these communities, the challenges of P2P-based approaches have to be investigated. This includes aspects like security and privacy, as well as incentivization models for community members to contribute to the infrastructure.

**RQ 2 - Scaffolding: How to support the creation of decentralized CIS with the help of Web-based tool support?** The unique properties of both CoPs and decentralized CIS open up new areas of research for supporting communities in developing for this new type of infrastructure. Scaffolding, in particular in the form of MDWE, allows to provide supportive structures for communities from the requirements engineering phase up to the success evaluation of the application.

**RQ 3 - Learning: How can learning communities benefit from using a decentralized CIS?** Finally, this third research question investigates the impact a decentralized CIS has on learning communities. We investigate how well both the development and the hosting of the infrastructure and the applications running on it are received. The question how these new approaches perform, and how lifelong learning communities benefit from them, is the overarching theme of this dissertation.

## 1.3 Thesis Contribution and Research Context

This thesis explores the opportunities and challenges decentralized CIS provide, with a certain focus on the design and deployment of backend microservices as their infrastructure. Following a design science approach as described in [HMPR04] and [PTRC07], we developed multiple artifacts, all individually demonstrated and evaluated, which together form the outcome of this dissertation.

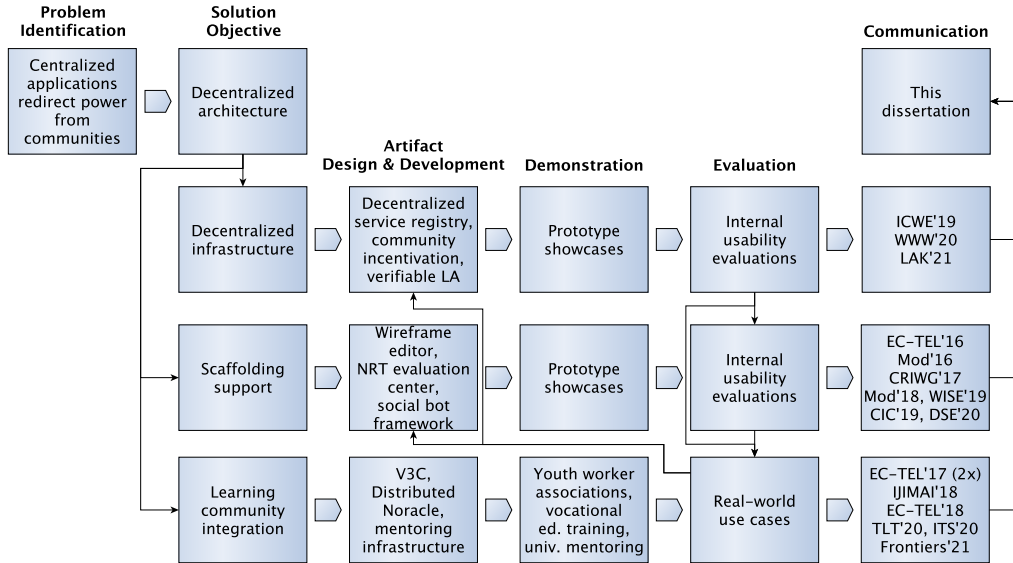


Figure 1.1: Overarching design science process of this dissertation.

Fig. 1.1 provides an overview on the overarching design science process we followed that lead to this dissertation. As one can see, the three main areas, decentralized infrastructure, scaffolding support and learning community integration build three separate iteration cycles, interwoven by requirements stemming from the progress of each individual cycle, and thereby influencing one another. To give a better impression on the developed artifacts, their connections and integration into an overall decentralized CIS approach, we provide such an overview with Fig. 1.2, sorted along the lines of the three main chapters.

The two chapters “Infrastructure” and “Scaffolding” (cf. Chapter 3 and 4) are depicted in the upper part, with the third chapter “Demonstration and Evaluation” (cf. Chapter 5) depicted in the lower part of the figure. From a scaffolding perspective, we produced and extended two main artifacts during the scope of

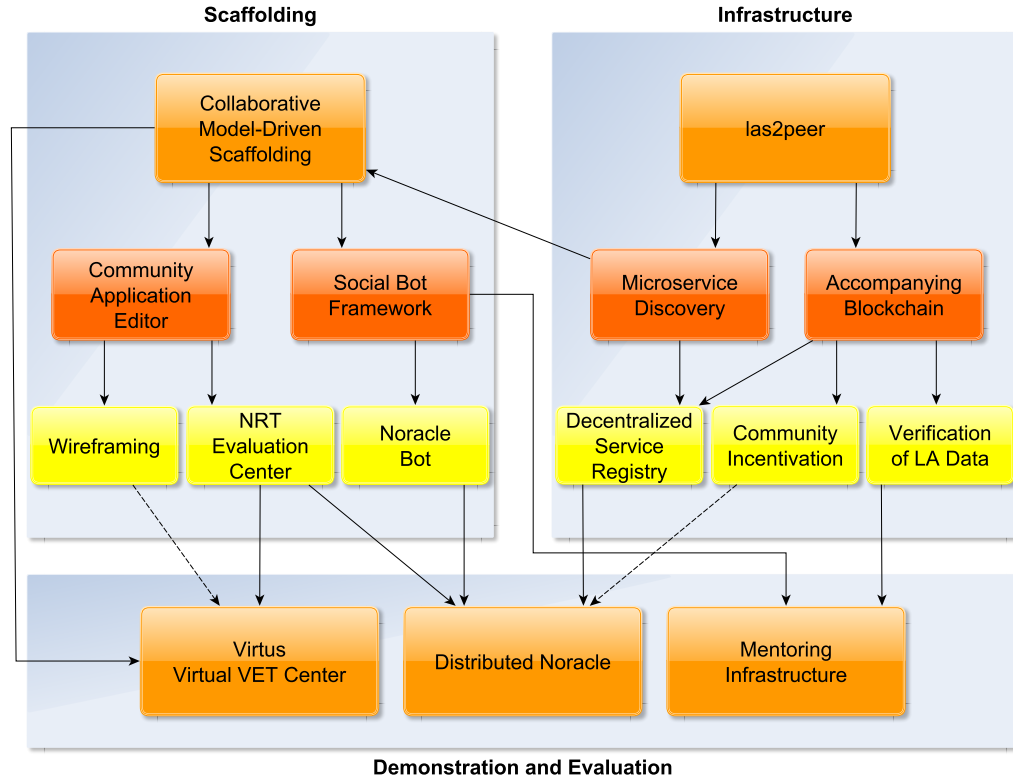


Figure 1.2: Overview on the artifacts produced in this dissertation.

this dissertation. The first is the *Community Application Editor (CAE)*, a MDWE environment with its two large extensions of collaborative *Wireframing* and the *NRT Evaluation Center*. The second is the *Social Bot Framework (SBF)*, of which the most prominent example introduced in this dissertation is the *Noracle Bot*. It has to be mentioned, that all these artifacts base on las2peer [KRLJ16], a decentralized, Open Source Software (OSS) environment for community-oriented microservice development, deployment and monitoring. For the “Infrastructure” part of this dissertation, las2peer not only builds the basis for technical development, but it also provides the conceptual grounds for undertaking the research in the domain of decentralized CIS infrastructures. Here, the main artifacts produced are las2peer’s *Microservice Discovery* mechanism, as well as the *Accompanying Blockchain*. Combining these two artifacts, we created the concept and implementation of a *Decentralized Service Registry*, which based on the conceptual findings of the

microservice discovery mechanism and uses the blockchain technology. As well, the *Community Incentivization* approach of las2peer is based on the blockchain, introducing L2Pcoin as the blockchain-backed community currency. Finally, the *Verification of LA Data* as the third artifact that makes use of the blockchain was developed. The microservice discovery mechanism also influenced our scaffolding research, with its ability to show the mash-up/interface compatibility of multiple services directly integrated in the CAE.

This dissertation features three large evaluations and demonstrations, all in different fields of the TEL domain (cf. Chapter 5). The virtual training center platform *Virtus Virtual VET Center (V3C)* blends Web-based formally-certified virtual training courses with self-regulated and social learning in synchronous and asynchronous learning phases. It uses the model-driven scaffolding techniques to design learning units on the Web, directly within the same platform as the training center itself. The resulting learning courses provide a Massive Open Online Course (MOOC) style learning environment that gets combined with informal social learning. The NRT evaluation center provides an integrated Learning Analytics (LA) approach to collect, store, analyze and visualize data for different purposes like certification, interventions and gradual improvement of the platform. The V3C was used within a certification program for vocational training in the domain of “Tourism and Hospitality” and “Social Entrepreneurship”. The second evaluation and demonstration was conducted in the domain of the European Voluntary Service (EVS), specifically in preparation and reflection workshops at the start and end of the participants engagement within the program. Here, we provide a decentralized infrastructure for knowledge building, together with an application developed on top of it, called “Distributed Noracle”. It helps the participants to reflect on their progress within the program, highlight open questions and facilitate their discussion. It is based on las2peer and uses both the decentralized service registry, as well as the Near Real-Time (NRT) evaluation center and the SBF, with the “Noracle Bot” being specifically developed for this application. Finally, our third and ongoing evaluation and demonstration is based within the domain of digital mentoring processes in university settings. Here the infrastructure, based on a fusion of las2peer and a kubernetes cluster is used to provide the verified LA data approach, which collects data from multimodal learning sources. We extensively apply the SBF within this demonstration to create various social bots that build the primary interface for both mentors and mentees. While we started evaluating this infrastructure already within multiple large university courses, it is an ongoing project and this dissertation only covers the first half of it.

## 1.4 Thesis Outline

This dissertation is structured as follows:

- **Chapter 2: Research Context** This chapter gives an overview on the context this dissertation was conducted in. We discuss related work on (decentralized) architectures for CIS, service registries and distributed consensus, as well as incentivization, scaffolding and MDWE. Finally, we provide an overview on the topics of (technology enhanced) learning we make use of in this dissertation, such as social learning bots and LA.
- **Chapter 3: Decentralized CIS Infrastructures** This chapter presents the decentralized infrastructure we developed. We start by describing las2peer, the technical foundation of our implementations, which we both used and extended during the course of this dissertation. Then, the microservice discovery and the community contribution incentivization are presented. Finally, we introduce the blockchain-based verification of LA data extracted from multimodal sources.
- **Chapter 4: Scaffolding Decentralized CIS** This chapter discusses the different approaches we took to enable the scaffolding of decentralized CIS with its two main contributions: the *Community Application Editor (CAE)*, a MDWE environment and the *Social Bot Framework (SBF)*.
- **Chapter 5: Distributed Learning in Decentralized CIS** This chapter describes the real-world application of the artifacts presented in the previous two chapters. In three evaluations and demonstrations, we apply and evaluate various parts of the decentralized CIS infrastructure and its scaffolding parts. We start with a large application in the domain of educational vocational training. The second demonstration is conducted in the domain of European youth workers, with a special focus on the EVS. We conclude this chapter with presenting a distributed mentoring architecture, based-on the contributions of this dissertation, which is used in student mentoring processes at universities.
- **Chapter 6: Conclusion and Future Work** Here, we summarize the contributions of this dissertation and reflect on the stated research questions. We conclude by providing an outlook for promising future work.



# Chapter 2

## Research Context

The term Community of Practice (CoP), coined by Étienne Wenger in the early 1990s [LaWe91], describes a group of people who share a concern or a passion for something they do and who interact regularly to learn how to do it better [Weng98]. Community Information System (CIS) research [Klam10] tries to provide CoPs with methodologies and tool support to fulfill these goals. In this chapter, we present the research context this thesis is embedded in, which can be roughly split up into the domains of *Human-Computer Interaction (HCI)*, *Web Engineering* and *Technology Enhanced Learning (TEL)*. Fig. 2.1 describes the interplay between these domains in the context of this dissertation.

In the following, we cover each of these domains with a specific focus on those aspects that built the background of this work. This includes research on decentralized architectures (Sec. 2.1), microservice registry and discovery (Sec. 2.2), distributed consensus (Sec. 2.3), incentivisation and trust (Sec. 2.4), scaffolding (Sec. 2.5), social bots (Sec. 2.6) and learning in CoPs (Sec. 2.7). With the overarching goal of this dissertation being the support of distributed lifelong learning CoPs, this chapter thereby lays the foundation for this work.

### 2.1 Decentralized Architectures

#### 2.1.1 Peer to Peer Networks

In general, Peer to Peer (P2P) describes a decentralized set of clients (nodes) that do not rely on a centralized management (self-managed). Since the approach neglects any central unit, it can be seen as the opposite of the common client-server

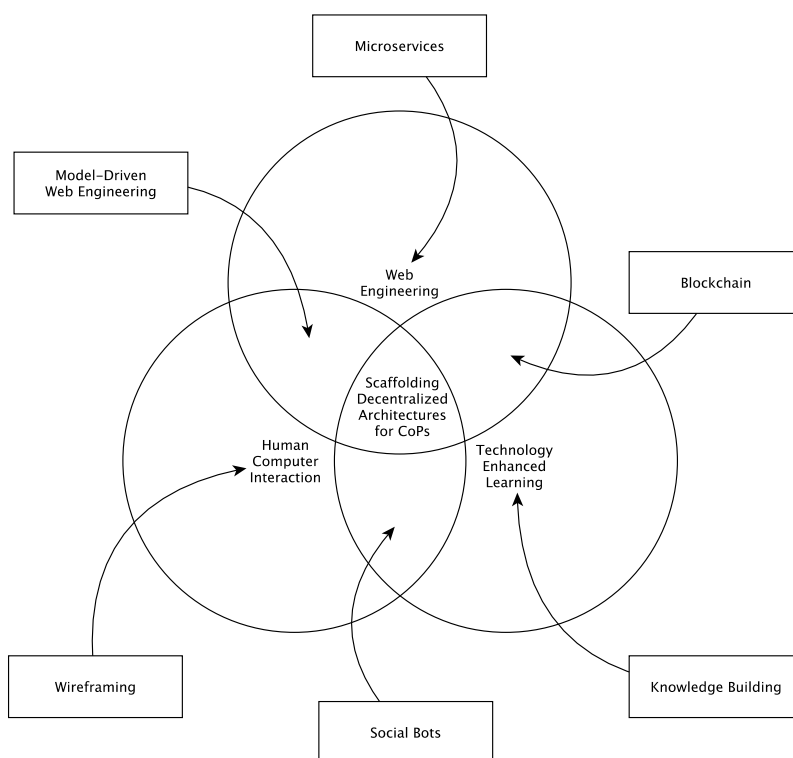


Figure 2.1: Venn diagram that shows the interplay between the three main research areas this dissertation is embedded in.

architecture. The information infrastructure [BBMR10] of a P2P information system consists of the physical infrastructure, the P2P storage overlay network and the P2P service overlay network.

As a result of the decentralization, it is necessary to give clients information about the network topology. This can be done in various ways. There are solutions where each node only knows a set of its neighbors as well as solutions where each node has information about the whole network topology. In this case, the term neighbors is referring to nearby nodes, based for example on distance or connection speed. The main advantage of the P2P concept is independence from a central server and therefore being more resistant to failure of single network entities.

P2P approaches can be divided into two categories: *structured* and *unstructured* networks. In the beginning of the broad use of the Web, P2P networks were mostly used for file sharing applications like eMule or KaZaa. These file shar-

ing platforms usually use an unstructured approach that allows for unconstrained placement of files within the network and a decentralized lookup scheme that only floods file requests throughout the network. While the advantages are based on the easier handling of the network, the main disadvantage is that it does not guarantee the finding of the requested (existing) artifact. Structured networks usually use Distributed Hash Tables (DHTs) to store node location information at each node [RoDr01]. In comparison to unstructured networks, the finding of artifacts is guaranteed within certain boundaries of network hops.

The ability to distribute tasks, which file sharing platforms make use of by splitting up the source of the file download to multiple nodes, makes the P2P approach also interesting in areas other than file sharing. The emergence of mobile terminals with sufficient speed to serve as infrastructure of information systems lead to P2P-structured, smartphone-powered “Mobile Hosts” [SJPr06], which coined the term “Mobile Web Services” (Mob-WS) [AHWa07].

The question of system maturity, flexibility and also interoperability has been and is still an active research area also in the domain of TEL [OcTe17]. The idea of using P2P -based information systems for sharing of educational resources came up first with the creation of EDUTELLA [NWQ\*02], a network for exchanging information about learning objects. Another active research domain is the use of P2P information systems for (decentralized) social networks (e.g. [NJM\*12, GAS\*16]), as the P2P conceptual architecture is closer to the actual nature of communication and collaboration in online communities [BuDa09]. Further development in this area is driven by the InterPlanetary File System (IPFS) [Bene14] project, which describes itself as a *peer-to-peer hypermedia protocol* and shares the concern for *increasing consolidation of control [on the Web]*, or the *The Invisible Internet Project (I2P)*<sup>1</sup>, which tackles the problems of privacy and data ownership. Related development approaches have been characterized as P2P cloud computing [BaMa14] and edge-centric computing [GME\*15].

### 2.1.2 Microservices

Microservices [Newm15] have emerged from the Domain Driven Design (DDD) approach in designing a system architecture. The DDD approach focuses on creating boundaries between business functions and model them into several separate models that are loosely coupled. By using this approach, a system can be divided into several small fine-grained services which are called *microservices*.

---

<sup>1</sup><https://geti2p.net/>

Each microservice works autonomously and serves one specific functionality. Microservices focus on modularity and decentralization of government, to ease scalability, availability and maintainability. On the frontend side, the concept of DDD applies to Web components. The approach focuses on encapsulation and separation of functionalities of frontend Web elements or Document Object Model (DOM) [OvSt15].

### Microservice Frameworks

Nowadays, microservices build on top of virtualization technologies like *Amazon Web Services (AWS)*<sup>2</sup>, build the basis for cloud architectures of big software companies. Examples are Netflix with their *Spring Cloud Netflix*<sup>3</sup>, Spotify's *Apollo*<sup>4</sup> or Twitter's *Finagle*<sup>5</sup>. Commonly, these main drivers of microservice architectures use container-based deployment. Although it has to be noted that these companies have released their framework as open-source, the resulting infrastructures remain mostly black boxes. Another downside of these environments is their reliance on provider-owned, centralized data centers, regardless of the service model they are based on. As ownership drives centralization, it is hard to imagine a business model for storage/computing capacity providers in a purely decentralized architecture, although projects like the *SAFE Network*<sup>6</sup> show how even this can be achieved. Existing approaches either focus on (open-source) microservices architectures in the cloud or on purely P2P-based architectures to enforce privacy and data ownership. In comparison to related research that mainly focuses on the redesign of monolithic applications into microservice architectures (e.g., [BHJa15]), we focus on lowering the entry barriers in community-based microservice development for and on the Web by introducing established software engineering methodologies, such as model-driven techniques.

## 2.2 Microservice Registry and Discovery

At its most basic, a *Service Registry* maps the name of a service to a network location (e.g., a Uniform Resource Locator (URL)). In other words, it is a *name*

---

<sup>2</sup><https://aws.amazon.com/>

<sup>3</sup><https://cloud.spring.io/spring-cloud-netflix/>

<sup>4</sup><http://spotify.github.io/apollo/>

<sup>5</sup><http://finagle.github.io/>

<sup>6</sup><https://safenetwork.org/>

*service* (cf. the *Domain Name System* [MoDu88]). However, this functionality arguably does not merit the term “discovery”, as the service must already be known by name and the service requester is hardcoded to its exact interface.

*Service Discovery* encompasses varying degrees of functionality, depending on the context. In its most basic form, it refers to the publication and lookup of the network location of a service which is already known by name in a registry (*service location discovery*). This registry may also allow the retrieval of services matching a formal description (*semantic service discovery* or *matchmaking*), and thus requires that services publish a machine-interpretable description of their capabilities. This meaning is central to the vision of the Semantic Web [BHLa01], in which data stored in potentially disparate sources (e.g., published in different formats, by different communities) can be automatically discovered, processed, and to some degree understood by machines [W3C13]. Finally, *end user service discovery* goes beyond programmatic discovery and aims to help users find Web services relevant to their interests, e.g., by employing *recommender systems* or the user’s physical and logical context [LMC\*16].

Most service discovery systems include a *service description* in the *service announcement*. In his description of the Service Location Protocol (SLP) in 1999, Guttman envisions this as “well-known attributes”, defined in *service templates*, forming a common “vocabulary” across vendors [Gutt99]. In the same year, Sycara et al. introduced Larks [SKWL99], an “agent capability description language”, which allows semantic matchmaking using inferences based on a local ontology. As example they give a simple subclass relationship, where a request for information on AirMissions yields a database of AWAC-AirMissions. In the following years, a multitude of related specifications were introduced (e.g., [CCMW01, UDDI00, ABH\*02, MPM\*05, AFM\*05]), but despite being the focus of extensive research, none of them gained widespread industrial adoption, presumably due to the steep learning curve. Instead, Web service developers embraced the RESTful paradigm, which – in line with a general trend toward lightweight, human-friendly standards – utilizes well-known and easy to use standards and tools [PZLe08] and in particular largely eliminate the need for defining communication endpoints and message exchange formats.

The concept of Representational State Transfer (REST), introduced by Roy Fielding [Fiel00], has some similarities to and is in principle compatible with the Semantic Web: both focus on resources and their relationships (expressed as links) [PRMa11]. By means of *hypermedia controls*, a RESTful API indicates all possible actions on the resource in question, allowing a client to dynamically discover them and automatically interoperate with the service. In theory, that is,

because just like Berners-Lee's, Roy Fielding's vision remains largely unrealized: The vast majority of APIs only follows a subset of Fielding's rules [RSK12, Fowl10], despite his emphatic insistence that they are essential and mandatory in their entirety [Fiel08].

For RESTful Web services, the service description method with the largest developer mindshare is the OpenAPI Specification (OAS) (formerly *Swagger*), which “allows both humans and computers to discover and understand the capabilities” of a service and specifically the generation of interactive documentation, client and server-side code, and unit test stubs [Open18]. OAS documents are written in the YAML Ain't Markup Language (YAML) format or equivalent JavaScript Object Notation (JSON), which makes them easy to read and write even with little technical knowledge, but nevertheless well-defined and machine-readable. This enables stakeholders to design and agree on the Application Programming Interface (API) ahead of its implementation, providing the authoritative single source of truth during all phases of development. Service matchmaking based on OpenAPI descriptions can help developers find compatible endpoints and facilitate component reuse in a microservice architecture.

A great variety of architectures for service discovery has been proposed. They are often classified according to the degree of centralization of the registry [Klus08, RKL09]. However, the most simple scheme is to not use a registry at all, but to propagate service queries or advertisements via flooding. It is clear that the communication overhead of this approach prevents it from scaling to large networks, instead it was suggested for home networks or even in cars, where the number of participating devices was presumed to be very small [Gutt99, BeRe01]. For medium-sized networks, a single central registration server may be used. The API of such a registry consists of *service publication* and *service lookup*. The registry simply caches the service description published by the *service provider* until some time-out is met and answers the *service requesters'* queries accordingly. While this approach can work well in controlled environments, several issues arise when attempting to serve large, geographically distributed, or heterogeneous networks: First, having a single registry server is neither fault-tolerant nor scalable. Further, if services should be accessible from across the globe, latency may be an issue. Finally, the registry is under the control of and must be maintained by a single entity. Out of these considerations emerged distributed service registry architectures, which can be classified into three domains according to the way they store service descriptions and state information:

1. *Replicating* [SLKe04], where registries attempt to have the same, complete

state.

2. *Distributed* or *Federated*, where registries only store information about local services, but forward queries about other services to a cooperating registry.
3. *Peer-to-peer* [TSNe03, Klus08], where information is also stored decentrally, but all participating registries use a common P2P protocol, negating the need for manually configuring and setting up sharing agreements between them.

Each of these is appropriate for certain use cases. Current commercial systems such as Netflix's Eureka<sup>7</sup> and HashiCorp's Consul<sup>8</sup> fall into the first two categories (or some hybrid combining both), with local registries assigned to each data center or region. P2P service registries have to our knowledge been primarily the subject of academic inquiry rather than deployed in practice. Most of them utilize a DHT, where service descriptions are addressed by the hash of their contents. However, unstructured P2P registries have also been proposed [Klus08]. In both cases queriability is a crucial issue, specifically the search capabilities of the P2P storage beyond exact match lookup and even completeness, i.e., the guarantee that an existing entry can be located. Much effort has been put into extending structured P2P overlays to allow attribute, wildcard-based and other advanced queries (e.g., [PSNS03, SGA\*05, SSDN02, ScPa04, YaZh04]), but these limitations remain a major obstacle.

## 2.3 Distributed Consensus

When we discussed distributed service registries in the previous section, we implicitly assumed that all nodes comprising the system are trustworthy, i.e., operating correctly without either accidental or malicious misbehavior. For corporate networks and many other use cases this is a reasonable assumption. But for a decentralized system that is open for anyone to participate in, as a peer among equal peers, a different approach is required. We use the term *open decentralized system* to denote exactly that: A system of autonomous peers, who may join or leave at any time, and whose goals may not align with one another. Further, there is no single centralized authority, which could coordinate or serve as a universally trusted entity in the system. A fundamental problem of such decentralized systems

---

<sup>7</sup><https://github.com/Netflix/eureka>

<sup>8</sup><https://www.consul.io/intro/>

is how to ensure that received information is up-to-date and authentic, despite being unable to trust any particular node [Matt16]. This is an instance of a *consensus problem*, which has been in the focus of distributed systems research since the early 1980s [PSLa80]. More recently the topic has come into the spotlight due to the apparent success of cryptocurrencies, which purport to solve the problem on a global scale. In essence, a functioning decentralized system needs to agree on a common state. The nodes of a distributed database need to agree on the contents and order of the applied operations, while cryptocurrencies deal with the specific case of tracking the participants' account balances. Thus a secure, scalable consensus algorithm lies at the heart of decentralized systems. This often requires independent modules to collaborate to agree on some shared data, to reach *distributed consensus* [PSLa80, LSPe82].

### 2.3.1 Consensus Mechanisms and the Blockchain

A blockchain is a mathematical structure that represents a growing list of data blocks which are cryptographically linked and signed. This architecture provides a decentralized data structure with a synchronization mechanism to continue reaching distributed consensus over additions and modifications to the shared resource. Blocks are added in a linear manner, meaning that each block has one predecessor and successor. Each block contains the cryptographic hash of its predecessor, the timestamp of its addition to the chain and a list of generic transactions. While reading values can be done without additional costs, introducing changes to the blockchain requires computational power to validate and add the requested transactions to the ever-growing list, which is technically represented by transaction fees. Participating nodes of the chain are mathematically incentivized to verify and maintain the network, in other words, mining nodes get rewarded with the transaction fees contained in the block they helped verify. Usually, the chain of blocks is implemented by means of a *Merkle Tree* [Merk79] to allow for efficient processing. Different levels of privacy to access the chain provide different benefits and drawbacks. The levels are categorized in two dimensions – public vs. private and open vs. closed. The first duality describes write access permissions, while the latter is concerned with data access rights. In other words, a public chain means there are no restrictions on who can add blocks, while an open chain means the data is publicly readable. Public chain providers need to concern themselves with content moderation and incentives, since the user-base is unknown, while private chain operators can rely on the fact, that they know who their users are.



### Types of Consensus Mechanisms

While the most prominent consensus mechanism by far is the Proof of Work (PoW), due to its usage in cryptocurrencies, there are a few others which we also want to present here.

**Proof of Work** The PoW consensus has been introduced in the 1990s as a system to combat spam or deter denial of service attacks [DwNa93, JaJu99]. The main idea of a PoW consensus relies on a service requester providing proof that some work was completed, typically through computations. A key feature of this scheme is asymmetry — while the work must be moderately hard on the requester, it should be easy to verify for the service provider. A major downside of PoW consensus is the large overhead generated by each transaction. One of the most influential technologies which made PoW consensus popular, is the blockchain-based *Bitcoin* [Naka08], a cryptocurrency which combats the double-spending problem of digital tokens. This technology is based on a PoW consensus protocol in which the voting power is proportional to the computational power of the node. Even though in Bitcoin’s implementation, the blockchain is only used to store monetary transactions based on the Bitcoin currency, the blockchain in general can store arbitrary types of data.

**Proof of Space and Proof of Resource** Another approach to solve distributed consensus is *Proof of Space* [DFKP15]. A participating user must show that she has legitimate interest in a service by allocating a substantial amount of disk space. This is similar to PoW, but uses storage instead of computation, leading to more energy efficiency, due to the general-purpose nature of storage. A variation of this mechanism is *Proof of Resource* [Maid14], in which the ability of a node to store data blocks is measured in a zero-knowledge proof, spanning bandwidth, computing power, disk space and online time.

**Proof of Stake** In Proof of Stake (PoS) systems, the creator of the next block of the chain is chosen via a combination of random selection and optionally coin wealth, or so-called node age, i.e., her “stake” in the system. While selection by wealth alone would lead to centralization, as the richest nodes would receive permanent advantages, age-based algorithms can distribute the selection among the oldest nodes, while adding a time window to disallow repeated choice of a single node [KiNa12]. A big advantage of PoS systems as opposed to PoW, is that

in stake-based proofs, miners always own the coins, while PoW miners potentially own none of the currency as they only add the transactions. However, a major downside of PoS is that nothing-at-stake miners have nothing to lose from voting on multiple block chains, thereby preventing consensus, as there is very little cost for working on several chains in parallel.

### Smart Contracts

*Ethereum* [Wood14] is an open-source project, which employs blockchain technology. As a so-called second generation cryptocurrency, it utilizes a PoW-based consensus scheme. But instead of being only used as a cryptocurrency (like for example Bitcoin), Ethereum sees itself as a general purpose platform for decentralized applications. This is reflected technically in the syntax of its transactions. These can include code in a Turing-complete, stack-based bytecode language, whereas the transactions in Bitcoin's blocks are deliberately less expressive. This allows Ethereum users to write and upload scripts to the network, whose functions can be invoked by sending special transactions. Such scripts are called *Smart Contracts*. This concept describes self-executing programs, which run in the network in a decentralized fashion, hosted at a specific address on the blockchain. Each deployed smart contract consists of its program code, a data store, as well as an account containing *Ether*, Ethereum's currency. When a transaction triggers a smart contract function, the miner that includes the transaction executes the code and includes the updated state in the new block. All other nodes must also execute the code in order to determine whether the new block is valid (includes the correct result of the computation). Given certain restrictions on the computing power of an attacker in comparison to the nodes behaving correctly, this approach provides an immutable history of transactions.

It is clear that this massively redundant code execution is expensive in terms of resources. Ethereum charges a dynamic fee based on the number of executed instructions, which must be paid for by the transaction's sender. If its funds are insufficient, the execution is stopped. It should be noted that the cost of executing smart contracts limits Ethereum's scalability (in terms of throughput), and there are numerous proposals to alter the Ethereum protocol in order to improve performance, including radical changes to the consensus system [Bute17]. It is thus not economically feasible to deploy computationally expensive code or to directly store large amounts of data. Nevertheless, it enables a robust basis for decentralized applications, which can use smart contracts as the backing storage for both their data and the app code, securely distributed among participating nodes

on the chain. Examples of smart contract applications currently in use include financial contracts, games of chance, and notary applications [BaPo17], which can largely be implemented with very simple program logic, while documents can sometimes be stored elsewhere (e.g., via IPFS [Bene14]) and securely referenced.

While smart contracts could be used to manage funds, this is usually done in form of an *ERC-20* [VoBu15] token contract, meaning that the generation of tokens in this context is based on minting, which cannot be directly used to pay for blockchain transaction fees. The difference between *minting* and *mining* cryptocurrency is analogous with the real-world usage of these two words. Mining requires a significant amount of resources (computational power, electricity) to “unearth” new currency. This cost of electricity gives the currency its backing, while limits on the issue rate make sure of its rarity, and thus value. In blockchain applications, mining describes the process of validation and verification of all transactions to be put in a block that is to be added to the current chain of blocks. In contrast, minting requires no additional resources and is effectively inflating some token value unrelated to the “cryptofuel” used for transaction fees on the blockchain.

## 2.4 Incentivation and Reputation

In the context of online communities, research on why users participate stems mainly from a socio-psychological perspective. Forte and Bruckmann argue, that a sense of authorship and the so-called *cycle of credit* are to be attributed to the motivation behind the majority of the contributions in the Wikipedia community [FoBr05]. A valuable aspect of this is credibility, which, even though it is a lengthy process to obtain, allows users to assume more central roles in the community. Indirectly, credit can be used as a reward mechanism to log the contributions of a user. In this context, the authors also mention that community leaders should rise from the community and that they should be voted in, based on the credibility they gained.

Smith et al. argue, that reputation in itself is a motivating factor for user contributions, as users will be inclined to increase their rank in the community [SmKo99]. Additionally, when users have the feeling that their contributions have an actual impact on the environment, they feel a sense of efficacy and their contributions will likely increase.

On the opposite side of the spectrum, one can assume that user actions are driven by altruism or simply need, meaning the individual or the community

requires the produced artifact to fulfill some action. An additional possible motivation is the commitment or attachment of the individual to the community, in which case contributions to the community are done because it is best for the community. Here, individual and collective needs are merged, even though complete devotion to a community is rare. Some contributions are also a byproduct of individual behavior and are shared only because they have already been produced in a different context, as redistribution has costs close to zero [SmKo99].

In the context of social psychology, discrete emotions are emotions such as joy, sadness, fear or anger [DPR\*04]. Each of these has theories connected with them, however the usage of fear is deemed most useful in the context of user persuasion. Based on this theory, a viable strategy would be to first spark fear by implying that some privilege would be taken away from the user, then to provide them with information on how to avoid this problem. This solution will then become more persuasive and will convince users to contribute to the community. However, the case of the now-closed Limewire showed, that denying users access to system resources when they do not meet sharing requirements caused the communities to be more exclusive and failed to motivate the users in the long run.

Vassileva et al. suggest a number of theories with background in social psychology that can be transferred from real life communities to virtual ones [ChVa05]. One of these is called reciprocation theory. When we ask another person to do something for us, we are expected to provide appropriate rewards for their favors. This means that users should benefit from a P2P community if they are expected to join, participate in, and contribute to it. A community which does not provide benefits that outweigh the costs of resources, time and energy to contribute, is not sustainable and will degrade over time [Butl01]. Reciprocation systems reward its users for participation. To improve the effectiveness of the motivating strategies, two key questions need to be considered. On the one hand side, user participation and contribution need to be measured and evaluated. This is due to the fact, that the amount of retribution depends on user activity. On the other hand, the type of reward needs to be deliberated. If users do not associate value with the provided rewards, they will not be stimulated to contribute. P2P file sharing systems like KaZaA and Torrent use reciprocation to motivate their users to share files, by rewarding them with a better quality of service. The more a user shares, the higher her possible download rate is. Since the benefits are proportional to the amount of resources shared with the community, users are compelled by this strategy to stay online, contribute and share files. It even motivates to be concerned about the quality of contributions, to maximize demand for them. However, an adverse effect can also be observed. Users sharing uncommon files receive low quality

of service and are discouraged from further participation, thus decreasing the diversity of shared files in the network.

### 2.4.1 Trust

Trust can be thought of as being able to rely on the actions of another party and thus is closely linked to reciprocity. Hawlitschek et al. identified three fundamental targets of trust [HTWe16]. The first is *trust towards peer* and describes which information about a peer is needed to build up enough trust to enter a transaction with her. Strategies to facilitate trust towards peers typically include user profile completion, identity verification, screening/vetting and reputation. The second target is *trust towards platform*. Research shows that trust towards the platform is more important for consumers than trust in individual peers [HoCh11]. In other words, users rely on the owner of a platform to provide a safe environment. The last target of trust is *trust towards product*. Transactions in a decentralized sharing economy often mean hiring a professional that will never be met in person, or buying an item without the ability to see it before the transaction is complete. Strategies like product photo verification or third party certificates to indicate skill can be employed here to enhance trust towards the product or service provided. The *Trust-Confidence-Distrust* model [GJK\*01] was one of the first approaches that implicitly models distrust as an own entity, instead of treating it as the opposite of trust. Both trust and distrust between two peers can exist simultaneously. In the context of P2P, trust defines how much people are willing to contribute without direct personal gain from these contributions, or even without the guarantee that their contribution will be used by the community.

### 2.4.2 Reputation Systems

A reputation system allows users to determine the reliability of their peers. The higher the reputation, the higher the trust. The history of past behavior of an individual in a community will heavily influence the expectation that peers have of future interactions [RKZF00]. An intuitive conclusion is thus the usage of reputation to differentiate the quality of service. Reputation systems can act as a central anchor of trust. For a reputation system to be valuable, peers should be able to verify each other's reputation without direct interaction [FLSC04]. One can approach this problem by making the log of related reputation transactions public. This allows to recalculate the reputation score, based on the weights coming from the amount of trust towards certain nodes in the system. Since

actions in the community usually require two or more actors, reputation attached to each transaction builds a Web of trust.

### Examples of Reputation Systems

A number of reputation systems have been proposed in literature, most notably:

**EigenTrust [KSGa03]** computes global trust values of peers by aggregating local trust values, based on the notion of transitive trust, similar to the *PageRank* [BrPa98] algorithm.

**PeerTrust [LiLi04]** evaluates a trust metric by measuring five factors: the feedback obtained by peers, the total number of transactions of each node, the credibility factor for feedback sources, the transaction context factor (critical vs. non-critical transactions) and a community context factor. A decentralized trust management service calculates each peer's trust value.

**VectorTrust [ZhLi13]** establishes the notion of a trust vector and trust transfer over a trust overlay network architecture. A trust propagation scheme called *Trust Vector Aggregation Algorithm* only requires communication between neighboring peers in the overlay network. Additionally, an adaptive time window mechanism takes malicious fluctuations in trust values into account to accommodate for peers suddenly losing reputation, making the reputation hard to build, but easy to loose.

**PRIDE [DeDa04]** employs a local database of verified peers as a caching mechanism, while a self-certification mechanism similar to SDSI [RoLa96] provides cryptographic verification by exchanging signatures along with a rating after each transaction. An IP-based safeguard is employed to mitigate the "liar farm" vulnerability against "ballot stuffing" or spreading false reputation. The fundamental security assumption is, that it is difficult for an attacker to maliciously generate a non-contiguous IP address range, so the system should increase the security distance between peers to decrease the probability that the opponent has a liar farm. This however decreases the accuracy of reputation.

**StackExchange** employs reputation to let their community moderate contributions [MMSF13]. Convincing peers that the provided contributions are legitimate will gain the users reputation, e.g., an up-vote on a question yields 10 points, while a received down-vote costs 5 points. A sliding time

window (a maximum of 200 points gained per day) prevents further abuse. At any time, users can see their progress in the reputation system and inspect the privileges they earned.

**GNS [WSGr14]**, the decentralized name system of GNUnet, securely and asynchronously manages and shares sets of user attributes. Access to user data in GNS is managed via identity tokens. The tokens are used for a Diffie-Hellman-based key exchange to obtain and orchestrate access tickets to user data. The resulting attribute-based encryption mechanism allows users to selectively grant and revoke access to sensitive data in the decentralized network. Attributes get queried based on the requested context, meaning that, e.g., a mailing list would be given access to the user's email, however not to more sensitive data like medical records. Such a personal query requires additional trust relationships to be formed prior to sharing. This is done by means of a privacy-friendly protocol, which does not rely on central servers to provide user identity management. GNS is used to authenticate servers in a decentralized network and essentially to supersede the currently prevalent Domain Name Server (DNS) infrastructure and the X.509 Certificate Authorities (CA) that come with it [SBSc18]. Abusive user patterns in GNUnet are detected by a supervised learning algorithm [GBGr16], combined with a protocol for privacy-preserving queries for the classification agent to be able to operate.

### Challenges of Reputation Systems

A fundamental assumption for reputation systems to actually provide meaningful insights is, that inputs and outputs of the system have to be contextually relevant to the field they are applied to [Farm14]. In other words, a credit score which measures financial aspects of an individual will most likely be a poor indicator of her potential benefit as employee or her taste in music [Cohe09]. An important factor to take into consideration is the focus on the quality of content, as opposed to its quantity. Solely rewarding the latter will lead to a significant drop in quality, in favor of empty or meaningless content generated in troves. Being exposed to peers casting possibly negative votes can make a daunting impression on new users, so an opt-in mechanism for the reputation system can be used to make the participation in mutual rating a conscious decision.

An interesting problem concerning rating and reputation systems is the so-called normative conformity – users subconsciously tend to conform to norms to

“fit in”. In other words, a well-received rating subject will likely continue to receive good reviews, as long as users trust in the collective choice of the community. This is not necessarily a bad thing, as social anthropologists argue that this concept is crucial to the formation of cultures [Hodg14]. However, while divergence is also a prevalent occurrence, human interactions should not be simplified to a duality between conformity and divergence, as even small children show behavior that follows a more dialogical interaction with their surrounding, as opposed to an individual learning experience.

## 2.5 Scaffolding

Coming from the construction domain, a scaffold is set around a building as a temporary supporting structure during its construction phase. As the construction site develops and the building gets completed, the scaffold gradually gets removed up to the point, that it is not needed anymore. The metaphor of using *educational scaffolding*, a term first coined in the 70s by Wood et al. [WBRo76], describes the process of teachers assisting students in their learning process. As students advance, the “scaffold” is more and more removed, such that students learn to work independently. During the last 20 years, the term has gained a lot of traction in the domain of educational research [HaGi05]. In the domain of computer science, this term can also be found and here it describes Web frameworks that help developers to reduce boilerplate code and to ease the maintenance of code bases across projects.

However, in this dissertation, we take the term scaffolding a step further by unifying the ideas from both the domain of education and computer science. On the one hand, we use scaffolding to describe template-based development scenarios for a decentralized CIS platform. On the other hand, we also interpret it in a more pedagogical sense and use it to describe the processes and best practices for CoPs to develop their applications. In the following, we present related work on *Model Driven Development (MDD)*, *Model-Driven Web Engineering (MDWE)* and *wireframing* as means of scaffolding in the development of CIS.

### 2.5.1 Usage in Digital Learning Environments

The utilization of MDD creates a model during the process, which generates an implementation via transformations. This procedure also focuses on the development of the respective model [WHRo14]. In software development, a conceptual



model is seen as an abstraction of the software. Schmidt emphasizes that abstractions of the solution are created instead of abstractions of the problem [Schm06]. Accordingly, the focus of the developers is on the modeling environment and the translation of a model into an executable solution, while domain experts are responsible for modeling and creating the solution. Using this system, different user groups such as developers or project managers can develop abstract models together.

The modeling of concepts is also conceivable for online learning domains, by using it to model concepts within MOOCs or webinars. For learning environments as the medium, the goal of individualized learning is to deliver “the right content, to the right person, at the proper time, in the most appropriate way” [ShTo03]. Through monitoring, the system should be able to interpret the learner’s activities based on a domain-specific model [PaLo04], such that it can infer the user’s requirements and preferences. Furthermore, the system interacts with the learner in a way that it facilitates the user’s learning process. In both variants of learning, the traditional face-to-face learning and e-learning, the three main factors are the *teacher*, the *content* and the *learner* [BeAd10].

To focus on scalable personalized teaching, we rely on Adaptive Learning Environments (ALE). Paramythis and Loidl-Reisinger describe the domain-, learner-, group- and adaptive model as the typical models encountered in virtual learning environments [PaLo04]. The domain model represents the course itself and can also contain information like workflows, participants, roles and the like. The whole knowledge of a domain is formed of a set of knowledge elements. Brusilovsky and Millán claim that prerequisite links between those knowledge elements are the most significant ones because they represent the idea of learning a related concept before learning another [BrMi07]. The learner model can be implemented in a different way depending on the ALE, but all learner models share the capability of monitoring the live status of a learner. In particular, the model has information about interaction time and stores the interaction history. Thus, the model has assessment information which can be used for analyzing the performance and to draw conclusions on upcoming learning steps of the student. Multiple learners can be clustered by the learners’ characteristics (e.g., behavior). These clusters are created dynamically due to their changing characteristics over time. To show the learner “what” to learn “when” and “how”, the adaptive model provides the subsequent instructions. During usage, the model can acquire the system’s behavior with the information provided by the content model. Through the model, the current progress of the learner can be monitored and the model can determine the subsequent instructions by analyzing the relationships between the knowledge elements. These definitions and learning flows can be a difficult and challenging task. As

one solution to this, Nicolaescu et al. presented SyncLD, a system that supports Web-based, NRT collaborative editing of “learning design” models [NDK113]. This system was built for CoPs and enabled end users to create learning flows.

### 2.5.2 Scaffolding of Community Information Systems

While there are multiple ways of introducing scaffolding support to CIS (service) development, this dissertation lays its scope on MDWE to lead and support community members throughout the development process. Closely related, wire-framing serves as a means to integrate less technical community members closer into the (MDWE) development process.

#### Model-Driven (Web) Engineering

Most MDWE approaches follow the philosophy of separation of concerns [Kent02]. Based on a comprehensive metamodel, certain views are defined to reflect specific aspects of a Web application. One of the first MDWE approaches that obeyed the separation of concerns idea in MDWE was OOHDM [ScRo98], with the goal of dealing with the increasing complexity of Web applications. It described a methodology for systematic guidance to design large scale, dynamic Web applications. The main activities of the OOHDM methodology comprised a conceptual, navigational and abstract user interface design and proposed how they are implemented in the final Web application. A slightly more recent, as well as ongoing, MDWE methodology is the UML-based Web Engineering (UWE) [KoKr02], which was conceived as a conservative extension of the UML. Thus, already existing concepts of the UML are not modified, the new extensions are just related to existing concepts. The first extensions are UML stereotypes, which are used to define new semantics for model elements, e.g. a navigation link. The Object Constraint Language (OCL) is used to define constraints and invariants for classes. UWE follows the separations of concerns principle to split up the modeling process into the conceptual-, navigational- and presentation modeling part. WebML is another MDWE approach developed in 2000 [CFBo00]. It does not propose another language for data modeling, but also extends the UML and is compatible with classical notations of ER-diagrams and others. WebML as well emphasizes the concept of separation of concerns. Therefore, the development process is divided into four distinct modeling phases. The structural model represents the content of the site expressed as UML class- or ER diagram, the hypertext model consists of a composition- and navigation model. The former one describes which entities of

the structural model are composed by a certain page and the latter one specifies the links between pages. The third one is the presentation model which expresses the layout and graphical appearance of pages. Finally, the personalization model defines user and/or user group specific content. In 2013, WebML emerged into the Interaction Flow Modeling Language (IFML) [BrFr14] and was adopted as a standard by the Object Management Group (OMG).

ArchiMate is an enterprise architecture modeling language [Lank13]. Although not directly a MDWE approach, it is relevant related work for our approach, because of its interpretation of the separation of concerns paradigm. It separates the content and visualization of the view. The main advantage of this is the usage of different visualizations on the same modeling approach and vice versa. The content of a view is derived from the base model and expressed in the same modeling concept. The visualization on the other hand can be completely different from the actual representation of the model. ArchiMate allows to define a set of modeling actions, that alter the content of the model. These modeling actions are mapped to operations on a specific visualization of the view. This additional abstraction level allows to define any sort of visualization, like videos or dynamic charts. In this contribution, we use this concept of view separation to map certain operations on the wireframing editor to operations on the modeling canvas, which alter the current state of the wireframing-, respectively the modeling view.

## Wireframing

In Web engineering, a wireframe is an agile prototyping technique to sketch the skeletal structure of a Web application [AABe10]. There exist a plethora of wireframing and mockup tools on the Web. We here exemplary introduce Balsamiq<sup>9</sup> (as one of the most used ones) and Mockingbird<sup>10</sup> (as it features NRT collaboration and is Web-based). The idea behind Balsamiq is not to build large and fully interactive prototypes, which take hundreds of hours to develop and may lead to costly refinements if something can not be realized as intended. Instead, Balsamiq follows a more rapid development philosophy. This has the advantage that developers gain experience and evaluate components of the wireframe directly on a very early version of the Web application, which can also involve end user feedback. This feedback is used to tweak the wireframes and the implementation process starts again. Therefore, Balsamiq offers only limited interactivity features

---

<sup>9</sup><https://balsamiq.com/>

<sup>10</sup><https://www.gomockingbird.com/>

on a wireframe. Mockingbird is a Web-based wireframing application that offers NRT collaborative editing. The graphical editor offers the most common User Interface (UI) elements of today's Web applications, which can be rearranged and resized freely on a page. Similar to Balsamiq, it is possible to link pages and preview them to demonstrate the Web application's interactivity flow.

Mockup Driven Development (MockupDD) is a hybrid, model-based and agile Web engineering approach [RRG\*11]. The main goal of MockupDD is to extract and combine the advantages of MDWE methodologies and the rapid collaborative design process of wireframing, to add agility to existing MDWE approaches. MockupDD describes a transformation approach from a mockup to a comprehensive model that is further transformed to the specific models of an arbitrary MDWE approach. In most related approaches, wireframes are not considered as models and their impact declines in later development stages. MockupDD tackles this with a generic approach to integrate mockups directly into the whole MDWE development process. An additional computational instance builds the bridge from the output of an arbitrary wireframing tool to an arbitrary MDWE approach. The MockupDD methodology begins by creating UI mockups with an arbitrary tool, e.g. Balsamiq or Mockingbird. The resulting mockup file is then parsed, validated and analyzed with regard to a Structural UI (SUI) metamodel, which denotes each UI control element, their compositions and hierarchical structures. The goal is to obtain a "sufficient enough" structural model of the UI. Based on this SUI model, another transformation approach to the specific model of the used MDWE methodology is required. To further enrich the representational strength of a SUI model, MockupDD includes a tagging mechanism. A tag is simple specification that is applied over a concrete node of the SUI model and consists of a name and an arbitrary number of attributes. The main purpose of a tag is to define functional or behavioral aspects of a certain UI element. It allows the designer to construct more complex wireframe specifications. A UI element may have an arbitrary number of tags assigned to it. A SUI model enriched with tags is also called a Structural UI with Tags (SUIT) model. The concept of MockupDD has been adapted to various modeling languages and domains (WebML [RRG\*11], UWE [RGR\*12], IFML [RiRo13], and specifically focusing on mockups of touch user interfaces [Ange16]).

## 2.6 Social Bots

Research of bots already started in the 1960s with the development of “Eliza” by Joseph Weizenbaum [Weiz66]. More recent research and especially developments of social bots in the domain of learning technologies started around the year 2005, with Fryer and Carpenter using chatbot technology to support the acquisition of language skills [FrCa06]. These early works of modern bot development were quite limited and only able to reply to specific questions with predefined answers. Kerly et al. used a chatbot to support the self-assessment and reflection of learners [KHBu07], where learners were able to discover and negotiate their own learner model by using the chat tool. In order to bring the relevance of bots closer to students, these bots were also integrated into introductory courses for CS students [Shaw12]. A team from the University of Edinburgh used a chatbot for one of their courses [Bayn15]. The bot replied to Twitter tweets, thereby taking the role of a teacher. The replies were based on keywords in the students’ request. As an example, the bot was able to answer questions regarding the submission deadline of assignments. The idea of this approach was to overcome possible shyness of students through conversation with a bot instead of a human teacher [BKCr18]. Krafft et al. used bots as virtual confederates, which behaved like human confederates in experimental situations [KMPe17]. In their work they gave four different ways to experiment on the bots. They proposed to randomize bot actions, attributes, behavior and the artificial which is created by multiple bots. Dibitonto et al. presented the design of LiSA, a social bot for Facebook which conducts surveys with students [DLTM18]. The bot interviewed the students mainly about the way they obtained information about the university.

The role of a mentor can be taken over by a chatbot, a software program conducting auditory or textual conversations. Natural Language Understanding (NLU) can be applied to analyze speech, and intelligent responses can be created by designing an engine to provide appropriate human-like responses. The results of a systematic literature review show that chatbots have only recently been introduced in education, but they offer opportunities to create individual learning experiences [WiSo18]. This can lead to an increase in learning outcomes and can support lecturers, as well as their teaching staff. Chatbots have also been extended in the field of mixed reality, which describes a spectrum between the real world and a purely computer generated world with the intermediate forms of augmented reality and augmented virtuality [MiKi94].

The XML-compliant language Artificial Intelligence Markup Language (AIML) [Wall03] consists of categories which contain a stimulus, or pattern, and a tem-

plate for the response. Categorical patterns within the stimulus are matched in order to produce the best response to the user's input. The potential of bots using this language was recognized relatively early and different dialog systems have been suited with AIML to enhance applications within the fields of medicine and education [MBL\*09, REB\*12].

Fischer and Lam introduced a social bot that asks the user questions to perform a medical diagnosis [FiLa16]. Here, it was evident that the chatbot was leading the conversation and yes/no questions limit the user with their response. AutoTutor [GCHO05] is another approach using predefined conditions for natural language processing in order to simulate a human tutor. CalmSystem [KEBu08] also included a social bot in the domain of e-learning which supports the meta-cognitive goals of self-assessment and reflection. With the presented concept, individuals can access their learning model and make inquiries via the chatbot.

Latham et al. introduced "Oscar" a Conversational Intelligent Tutoring System (CITS), which utilizes a conversational bot that adapts to the student's learning style [LCM\*10]. Lundqvist et al. used an SQL database as storage for data structures in combination with *OwlLang*, a script language for injecting knowledge from ontologies into AIML, for their implementation of an AIML chatbot [LPWi13]. The presented chatbot was used as a survey tool inside a learning platform to collect data on the quality. More recent approaches, such as the social bot introduced by Lim and Goh [LiGo16], are still based on AIML. The bots mentioned until now are categorized by Barbar et al. as retrieval-based bots [BLer17], while the following, more recent bots are considered as generative bots which are based on deep learning technologies [DeYu14] and the content is generated based on the context of the users' inquiry.

Varghese and Pillai propose a system for closed domains where they use a Recurrent Neural Network (RNN) for generating the response as a law assistant [VaPi18]. The inclusion of non-technical users can be done for example with SOCIO [PGLa18]. Perez-Soler et al. present a system where users can make collaborative requests and the system returns the corresponding class diagram after each change [PGLa18]. The collaboration of the participants and the recording of the requirements takes place in an Online Social Network (OSN).

## 2.7 Learning in Communities of Practice

This section introduces two important aspects of learning in CoPs, namely monitoring of LA data and knowledge building. While we make heavy use of the

former in this dissertation to gather the data needed in all three applications and evaluations of Chapter 5, the latter is of special importance for our second case study of the Distributed Noracle application in Sec. 5.2.

### 2.7.1 Monitoring of Learning Analytics Data

Vocational training and technological support for CoPs adapted to the digital, virtual domain not as fast as e-learning conducted in higher education at university level for example. Although its potential was first mentioned already at the end of the last century [Scha97], its adoption is still in its infancy. The result of this is that studies performing LA [Ferg12], to the best of our knowledge, consider either university/college or schools as the educational setting [NDK15]. From a technological point of view, most LA approaches use data generated by MOOCs created by open source course management system, with the most famous one being *Moodle* [DoTa03].

Using standardized data formats [BSFM13] to apply LA on, like the Learning Record Store (LRS) or the eXperience API (xAPI) standard [KeRy16], is an active research topic. The LRS is a server responsible for receiving, storing and accessing learning activities. It can be operated as a stand-alone system, but it can also be integrated into a Learning Management System (LMS). In the latter case, the LMS takes over the reporting and analysis using the data of the LRS, while the LRS connects to other activity providers. It also provides the possibility to connect to another LRS for further analysis of data from other sources. A LRS builds the basis for an xAPI ecosystem, an OSS specification of a data format for learning data. Any interaction from a tool via the xAPI is done through the LRS, allowing the system to store and retrieve xAPI statements. Those statements contain information about the actor, verb, and object. When an actor interacts with the tracked system, the verb describes the type of activity and the object describes what is being interacted with. In addition to these predefined data fields of the xAPI, several extensions are available to allow a highly customizable storage.

Having a well-defined data (exchange) format allows for an easier handling of the data to be analyzed, but it does not solve the question what data needs to be collected to understand past and predict future students behavior and react accordingly. Techniques like Educational Process Mining (EPM) can be used to retrospectively derive meaning of collected data logs. Bogarín et al. have used this technique together with clustering to predict success rates of students taking a Moodle course [BRCS14]. Click-based analysis is also very often applied to LA in Virtual Learning Environments (VLEs), like for example the authors of [WZNP13]

used the General Unary Hypothesis Automaton (GUHA) method, a data mining technique, to predict student drop-out rates. In general there can be concluded, that most studies consider the prediction and improvement of assessment results, activities and drop-out rates [SKD\*14] as their main goal. While the first two are of equal importance in a Self-Regulated Learning (SRL) context, the drop-out rate problem is more bound to the MOOC context.

### 2.7.2 Knowledge Building

This dissertation focuses on the support of CoP learning processes in the digital space. We understand it as a social process that involves negotiation of meaning and social construction of knowledge. With respect to learning as a social process, we apply the CoP concept to describe the emergence, transfer and preservation of knowledge [Weng98].

Question asking is seen as one of the most important skills for knowledge building, contributing to lateral thinking and better problem solving [Sloa17]. Question-based dialog is viewed as a specific type of a sense-making tool that is also representation-centric [MPOR18]. To help structure discourse analysis, computational linguistics has offered frameworks to examine collaborative sense-making in virtual environments [IQLB16]. For example, argumentation platforms offer a representation-centric approach to collaboration. Contributions are visually represented, categorized as issues, claims, premises and evidence, with modifying functions to support or refute other constituents of the argument. Cohesion graphs of discussion threads, which represent contributions as nodes at different levels, can examine lexical chains in discourse analysis to understand influence on conversation and identify key issues in conversation. Related work in this domain mostly deals with the issue of how face-to-face scenarios differ from online discussions and how to aggregate community knowledge [Meye03]. Instead of representing *knowledge* in the form of arguments, in the scope of this dissertation, we built the Distributed Noracle (cf. Sec. 5.2) to examine the *gaps* in community knowledge in the form of questions.

In the domain of school education, a theory that specifically focuses on social configuration for knowledge creation is the theory of *Knowledge Building* by Scardamalia and Bereiter [ScBe06]. The rationale behind it is that the knowledge called “state-of-the-art” is the sum of the knowledge of the community. Knowledge work therefore is the advancement of the state of knowledge within a CoP. Knowledge building explicitly focuses on the community knowledge advancement and stresses the temporary nature of ideas and theories. Every idea is improvable and every



theory can be refined, redefined or replaced by a new improved theory. To work on ideas, knowledge building uses a form of discourse that can be characterized as a cooperative process where participants are committed to progress, seek common understanding and expand the base of accepted facts. Knowledge building assumes that learners' understanding is emergent and that the development of complex cognitive structures for complex concepts is achieved by self-organization: "new conceptual structures [. . .] emerge through the interaction of simpler elements [. . .]" [ScBe06]. This is also applicable to knowledge of ignorance, which can rather be expressed by questions than by idea statements.

Coming from the field of organizational studies and knowledge management, the *SECI model* developed by Nonaka and Takeuchi [NoTa95] and its adaption to the Web 2.0 [CKJA07] describe the process of knowledge creation in four cyclic steps:

1. **Socialization** (tacit to tacit): the process of sharing tacit knowledge by collaboration and practice, through which learners develop a shared mental model.
2. **Externalization** (tacit to explicit): make this knowledge explicit, e.g., by writing it up, revealing the tacit knowledge.
3. **Combination** (explicit to explicit): combine explicit knowledge sources to create new knowledge.
4. **Internalization** (explicit to tacit): by using the explicit knowledge sources, the knowledge is internalized.

For emergent knowledge, revealed ignorance plays a pivotal role in both the theory of knowledge building [Scar02] and the SECI model (here especially in the "externalization" step, where both knowledge and ignorance can be revealed). The learning process of Inquiry-based Learning (IBL) starts with a question or statement of curiosity, sometimes called the "wonder moment" [STKS15]. Once an unanswered question is asked within a community, it challenges the ideas and theories of the community. A collective model of community ignorance results from the subsequent discourse.



# Chapter 3

## Decentralized Community Information Systems Infrastructures

### Summary

This chapter lays its focus on the development of the decentralized infrastructure used as a platform for the realization of the artifacts created in the following chapters. We present the decentralized service registry and its application in both service verification, community contribution incentivization and LA data verification.

**Contributions**  $\Rightarrow$  RQ 1. *Keywords:* Decentralized Infrastructure; Service Registry; Blockchain; Incentivization; LA. The results presented here have been partially published in [KRLJ16, LFGK17, LJK119, LGFK18, LGF\*20, LSK120, LBNK21]. This chapter contains partially information and content extracted from these publications.

In this chapter, we present the decentralized CIS infrastructure we developed during the course of this thesis work. It builds the basis for both the work presented in Chapter 4, as well as the applications and evaluations presented in Chapter 5. We start by presenting the design science process followed for the contributions presented in this chapter (Sec. 3.1), before we introduce the concepts and technical foundations for this chapter. Here, we present *las2peer* as the technical foundation, which both builds the technical basis of this thesis' implementation part, as well as it is itself treated as artifact that was extended and evaluated throughout this work (Sec. 3.2). The chapter continues with the first

iteration of las2peer’s microservice discovery implementation in Sec.3.3, before we introduce the *decentralized service registry and discovery mechanism*, based on blockchain technology (Sec. 3.4). Basing on this, we present the *decentralized community contribution incentivization* in Sec. 3.5, and conclude this chapter with a description of the *blockchain-based verification and consent management of LA data* (Sec. 3.6).

### 3.1 Methodology

Our design science process for this chapter is depicted in Fig. 3.1. It consists of five iterations. Our initial motivation for this research was the problem of the lack of microservice discovery for decentralized scenarios. Without them, service discovery in terms of browsing and also verification is not possible. We began tackling this issue by first developing a microservice discovery with API metadata and information taken from our MDWE scaffolding support (see also Chapter 4). The main evaluation results pointed to the fact, that we should tailor the service discovery (also) more to end users, additional to the still standing need to decentralize the technical side of the service discovery mechanism. Stemming from this, and with the knowledge gained from the initial internal user evaluation, we developed a decentralized service registry, based on blockchain technology [LJK119]. Our evaluation was based on an artificial community setting and also integrated technical evaluations, with regard to blockchain computing power and storage consumption. The outcome from an end user’s point of view was the *Service Explorer*, which we then evaluated in the next iteration step in one of our real-world case studies [LGF\*20] (see also Chapter 5). With this, we were able to prove the usefulness of the service registry for service discovery in decentralized learning settings. Still, a standing problem was the need to pay for transaction costs. Additionally to hosting a service, the operator of a P2P node had to pay with computing power for service announcements on the blockchain for the service to be visible and verifiable throughout the network. Our solution to this was monetizing the mined blockchain-blocks and to develop an incentivization system on top of it. With this, it is possible to reimburse service developers and hosters according to their contribution to the infrastructure, as well as it introduced a verifiable, reciprocal user- and service rating mechanism, which is used to increase the trust in community contributions within the decentralized network, based on their community ratings. Results of our evaluation were communicated in [LSK120]. Our final contribution of this chapter is closely coupled to the third application and evaluation presented

in Sec. 5.3 and introduces the verification of LA data, collected from multimodal data sources [LBNK21], via the blockchain mechanisms developed in the previous iterations of this design science methodology.

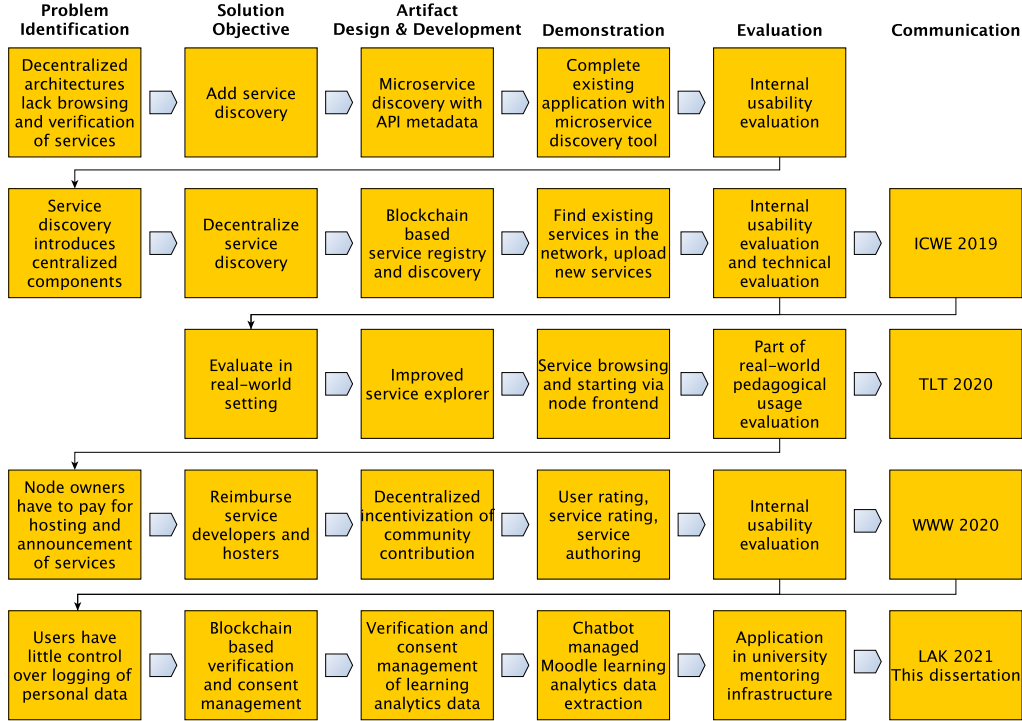


Figure 3.1: Iterations of the design science process we followed in Chapter 3.

## 3.2 Concept and Technical Foundation

One of the main themes of this dissertation is the decentralization of CIS for several reasons, both already spoken of in the introduction’s exemplary use case, as well as in the motivation for every single design science iteration described in this chapter. Conceptually, this means a shift from previous CoP-supporting infrastructures, which envisioned a centralized metadata repository at the heart of the CIS [Klam10]. Centered within the triangle of *CoP*, *Practices* and *Technologies*, this main “source of truth” retained the evolving community’s “long term memory”. In a decentralized CIS, we replace this centralized repository with decentralized microservice infrastructures. Instead of relying on a single repository,

the data is now distributed among the community’s network of knowledge-sharing nodes. The knowledge is not stored in a (custom-tailored) database anymore, but lives within the microservices. In a way, this shift from a metadata repository to a shared microservice infrastructure can be viewed as putting the metadata to action. This shift from a monolithic CIS to a decentralized approach transfers as the overarching idea throughout this chapter. Fig. 3.2 depicts this transformation.

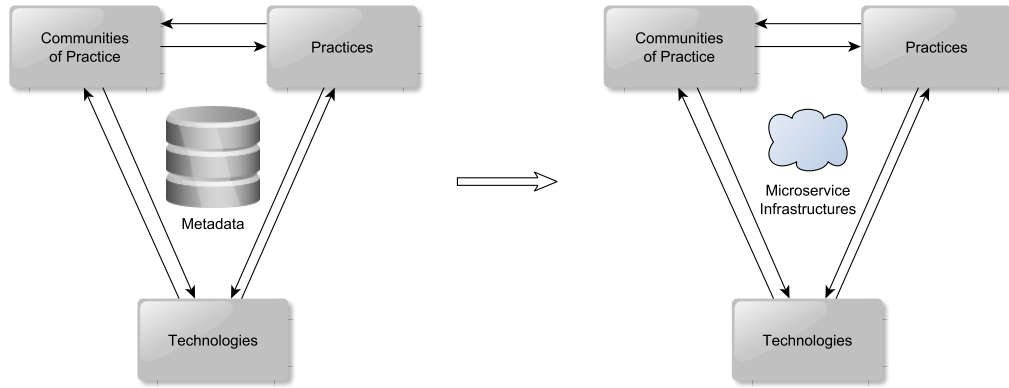


Figure 3.2: The transformation of a monolithic CIS ([Klam10], left) to a decentralized CIS (right).

To technically achieve this, we base our work on *las2peer*, an open source P2P framework for implementing and hosting Java microservices. A high-level description of the P2P architecture of *las2peer* can be seen in Fig. 3.3. As it can be seen, *las2peer* spans up a network of individual nodes, connected to each other, organized as a structured P2P network. Every acting entity of *las2peer* is an agent. Different types of agents exist for services, users, groups and more. Conceptually, all communication of *las2peer* takes place between these agents. E.g., if a developer deploys her service on a node, *las2peer* will register it as a service agent, if a user registers at a *las2peer* node, she is treated as a user agent. This way, the communication between a user and a service is conceptually handled the same as communication between services or users.

Every *las2peer* node consists of at least two components. The first is the *Distributed Storage*. This storage is partitioned and partly duplicated throughout the network, allowing for a shared, yet synchronized data store. Technically, we base our storage and inter-node communication mechanisms on Pastry [RoDr01], a P2P overlay network that provides both a messaging system as well as a DHT storage system. To ensure privacy, security and data protection, we added end-to-

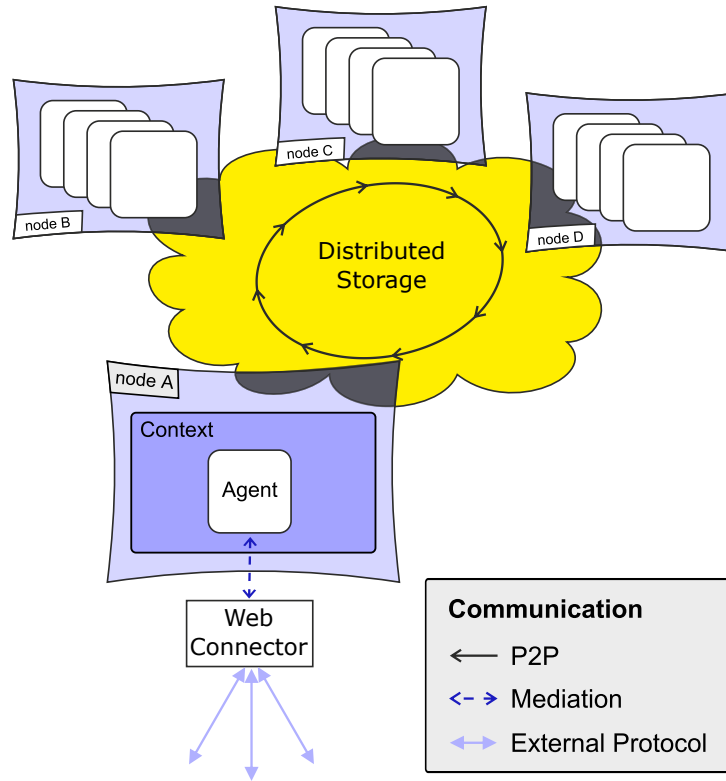


Figure 3.3: las2peer basic architecture [KRLJ16] (adapted).

end encryption in form of an *Envelope* system on top of it, ensuring each message and all data stored on the infrastructure is encrypted. The second component a node has to integrate is the so-called *Web Connector*. It realizes the communication to the outside, with the capability of routing RESTful calls to an application's (gateway) interface.

Our framework is capable of load balancing requests to microservices in the entire network, may it be because the service simply does not exist on the local node, or the node is currently overloaded with requests and offloads the task to other nodes in the network. Upstarting services register themselves to the network by calling a specific routine of the node, which then manages their location in the distributed storage for all nodes to look-up. This *Sidecar Pattern*-like [Newm15] service registration and discovery ensures that a connector will find the nearest service that currently is flagged as being capable of taking requests.

The communication between microservices is realized using a Message Ori-

ented Middleware (MOM) [BCSS99] that is based on the *Publish & Subscribe Pattern* [EFGK03]. Each node registers all running services as subscribers to their corresponding “Service Topic”. If a service wants to call another service, it performs a Remote Method Invocation (RMI) that is sent throughout the network. A node hosting a corresponding service that receives this request will route it to the service, which will handle it. The answer is then sent again in the same way throughout the network. Several timeout mechanisms and an acknowledgment system prevent messages with missing receiver to be forwarded endlessly or messages being answered by multiple services. By using the P2P network to enforce an Event-Driven Architecture (EDA) of microservice-based applications [Rich15], we target the needs of fast-changing topologies in CoPs, where complete knowledge of the network might both not be available or even desirable. Nodes can join and leave the network at any time, and the network keeps a persistent distributed storage with *Eventual Consistency* (following the BASE model of modern cloud computing architectures [Prit08]), regardless of the current topology. Besides this, it is of course possible for a microservice to implement and maintain its own database, separately of the distributed storage.

las2peer features its own logging, monitoring and evaluation suite, called *Mobile Community Information System Oracle for Success (MobSOS)*. Each node in the network monitors local events and sends them (encrypted) via the P2P network to a trusted central data collection sink, called the *Monitoring Node*. This monitoring node, in itself no different to other las2peer nodes, contains two services to handle data processing and provide data provision via a Web frontend. It not only features monitoring data, but also provides *community service success measurement*. A detailed description of the extensions made to the evaluation suite will be discussed with the introduction of the *NRT Evaluation Center* in the next chapter (see Sec. 4.3.4).

### 3.3 Microservice Discovery with API Metadata

While las2peer featured the possibility to browse services running on the local node, as well as services running on nodes known to the local node, there existed no registry or discovery mechanism for users to browse available services in the network. This issue is not limited to las2peer, but recognized as a general problem of decentralized systems based on DHTs [DaMa09], as they usually lack ranged queries. As a first step to tackle this identified problem, we implemented a microservice discovery system, based on API metadata.



### 3.3.1 Motivation

Our initial use case is connected with the MDWE environment we developed for scaffolding decentralized CIS (see also Chapter 4). Here, CoPs develop and mash up new applications, and deploy them within a network of las2peer nodes, decentrally and directly form the Web-based environment. As today’s Web applications usually are not developed independently, but are tightly integrated within an ecosystem of existing services, which they use and collaborate with [LZHS14], it is of high interest to the community, which existing services can be leveraged in the creation of the new community application. While our scaffolding environment included a basic service browsing mechanism to select the microservices that should be included in the application to be created and deployed, it lacked two crucial features.

1. *Service Metadata*: It did not contain any additional information about the services that could be used. Thus, the community had to be already knowledgeable about the service interface, origin or functionality, which renders the usage of externally developed services more-or-less infeasible.
2. *Deployment Information*: While the idea of deploying the complete infrastructure for each application by the community itself might have their specific use cases and can sometimes be desired (see also Sec. 3.4), information about already deployed services can help the community to make use of these deployments to not spend additional resources on re-deployment.

With these two shortcomings identified, we developed a microservice discovery mechanism that is based on API metadata<sup>1</sup>, collected by las2peer’s service scaffolding environment CAE. It is capable of visualizing the interface of microservices, using the OpenAPI specification. It also stores metadata of a service’s deployment status within the decentralized architecture. Finally, it provides a CoP with a first overview about the connectivity potential of microservices, by matching the collected API documentation of two services and calculating possible interface compatibility on a technical level.

### 3.3.2 Concept

Our concept is divided into two parts. First, we generate service metadata, by using API metadata and information taken from the service scaffolding environment.

---

<sup>1</sup>The prototype implementation was partially supported by a master thesis [Ceci18] under supervision of the author.

This is persisted alongside information about the current deployment of a service. Second, we use this metadata to provide users with information about available services running in the network. This also includes information about service interfaces and their possible connection and mashup potential for users to judge, which services work together or have the potential to easily be connected to each other.

### Generating Service Metadata

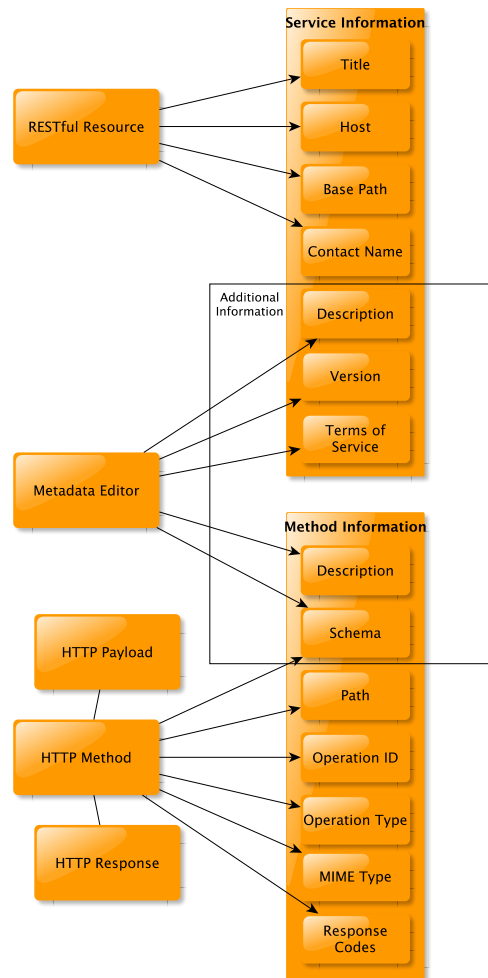


Figure 3.4: Metadata extraction from service model and metadata widget.

We extract the service metadata from two sources. The first is the service's model of the CAE, specifically the *RESTful Resource* and the *HTTP Method* with its connected *Payload* and *Response* objects. The second source of information is the *Metadata Editor*, which we developed to gain additional information that could not be extracted directly from the model. Fig. 3.4 provides an overview on the data extracted from the model and metadata editor. It is split up into service and method information. While the former provides global information valid for the whole service, the latter is collected for each method – and thus for each RESTful endpoint – of the service.

The metadata editor also contains a *Schema Editor*. Its purpose is to manage in- and output *Schemas* (according to the OpenAPI 3.0 standard<sup>2</sup>), which are used for the metadata extraction, OpenAPI specification creation, service matching and code generation. Since these are specific to a single RESTful endpoint, they are provided in the method information section, although implementation-wise they are managed on a service level for re-usability purposes throughout different methods of the service.

### Service Matching Support

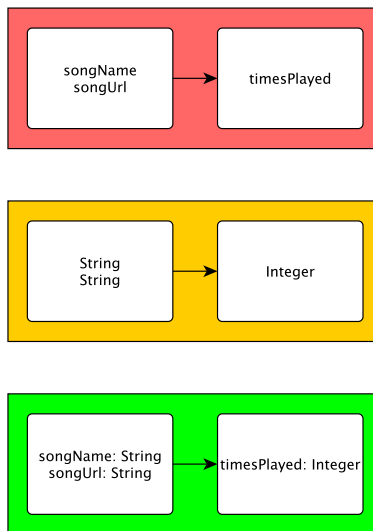


Figure 3.5: Exemplary matching levels of two service endpoints.

<sup>2</sup><https://swagger.io/docs/specification/data-models/>

We use the collected information described in the previous section to not only show community members the services available in the network, but also to provide information about their API interface and matching level in terms of interface compatibility. We decided to use a rather simple, yet intuitive traffic light system to indicate the matching potential for each RESTful method of a service with another one. Fig. 3.5 provides an example of the three different matching levels. Here, the red matching level would be shown for two services' methods whose OpenAPI schemas match only in names, but apart from that are both different in the amount of parameters and parameter types. The example here shows that the calling service' method provides the correct schema names and also consumes the correct output of the service to be called, but the schema's data types do not match. A yellow matching level indicates compatibility in data types, but the schema names of both methods do not match. While this technically does not pose a problem, it could indicate an "accidental" technical match that might semantically not provide the desired output. A green match finally indicates a matching both the schemas' name and data types. Here, the potential of two services of being capable of working together is rather likely.

### 3.3.3 Realization

We implemented this first iteration of a microservice discovery system as a *las2peer* service with access to a relational database that persists the service metadata. This *Service Registry* service, integrated with the CAE (cf. Chapter 4), connects to the various microservices that make up the functionality of the scaffolding environment to extract data from models and the metadata editor. Services developed and deployed from within the CAE are registered with this service, and the information is available to all community members from within the scaffolding environment. The resulting implementation can roughly be split up into its two functionalities. The first is the *Metadata Collection*, which is done by using the *Metadata Editor* and the information extracted from the CAE microservice model. The second is the *Microservice Discovery and Matching*, which is implemented by the *Deployment Viewer* and the *Service Matching Viewer*.

Fig. 3.6 shows a screenshot of the metadata editor. As one can see, it consists of the "basic information" section that collects additional data from the service developer, as well as the aforementioned schema editor. This schema editor allows for creating arbitrary schemas according to the OpenAPI 3.0 standard, which can then be used across multiple HyperText Transfer Protocol (HTTP) payloads or responses of the service API. Schemas and basic service information are then

The screenshot shows a web interface for editing metadata, divided into two main panels: 'Basic Information' and 'Schemas Information'.

**Basic Information Panel:**

- Component Name:** Music Service
- Description:** Service used to load and store songs.
- Terms of Service:** MIT License

**Schemas Information Panel:**

- Current Schemas:** A list containing 'Song'.
- Save Schema / Delete Schema:** Buttons to manage the current schema.
- Schema Name:** A text input field containing 'Song'.
- Schema Properties:**
  - Selected Schema:** A table listing properties and their types:

| Property Name | Property Type |
|---------------|---------------|
| title         | string        |
| album         | string        |
| favorite      | boolean       |
| numberOnAlbum | integer       |
  - Property Name:** A dropdown menu with 'Property Name...' as the placeholder and 'Integer' selected.
  - Save Property / Delete Property:** Buttons to manage individual properties.

Figure 3.6: Screenshot of the metadata editor.

persisted in a relational database for later reference.

The screenshot shows a web interface for viewing deployment information, featuring a 'Swagger Editor' header and a main content area.

**Microservice Information:**

- Component:** uat-testing-microservice-music
- Status:** Deployed
- Deployed since:** 3 days 16 hours 17 minutes 1 seconds

**Swagger Editor:** A green header bar with a 'Swagger Editor' label and a menu with options: File, Edit, Generate Server, and Generate Client.

**API Endpoints:**

- POST /post postMusic**
- GET /get getMusic**

**Parameters:** No parameters.

**Responses:** Response content type: application/json.

**Code / Description:**

| Code | Description        |
|------|--------------------|
| 200  | Response get music |

Figure 3.7: Screenshot of the deployment viewer.

Fig. 3.7 shows a screenshot of the deployment viewer. It displays both informa-

tion regarding the current deployment status of a service, as well as its API, either directly from the deployed service, or, if the service is currently not deployed, from the stored metadata described in the previous section.

|                                  |            |      |                                 |                    |   |
|----------------------------------|------------|------|---------------------------------|--------------------|---|
| uat-image                        | /get/      | get  | getPayload - (image)            | 200 - (image)      |  |
| uat-music                        | /post/     | post | postPayloadMusic - (musicImage) | 200 - (musicImage) |  |
| uat-music                        | /get/      | get  | musicId - (string)              | 200 - (musicImage) |  |
| uat-testing-microservice-image   | /postImage | post | payloadPostImage - (image)      | 200 - (image)      |  |
| uat-testing-microservice-image   | /getImage  | get  |                                 | 200 - (image)      |  |
| uat-testing-microservice-image-2 | /get       | get  | testPayload - (test)            | 200 - (test)       |  |
| uat-testing-microservice-music   | /post/     | post | payloadPostMusic - (imageMusic) | 200 - (imageMusic) |  |
| uat-testing-microservice-music   | /get/      | get  |                                 | 200 - (imageMusic) |  |

Figure 3.8: Screenshot of the service matching viewer.

Finally, Fig. 3.8 shows a screenshot of the service matching viewer. Once a user selects a certain microservice in the CAE’s “Microservice Select Widget”, the service matching viewer displays all RESTful endpoints of all services in the registry that provide a possible matching. From left to right, it provides the matching service name, the path and type of the matching service method, the method that matches the one of the currently selected service, its default return status code and schema type, and its matching level.

### 3.3.4 Evaluation

We conducted a small user evaluation, mainly to test the functionality of our approach and gain first feedback on the adaption of the microservice registry in general.

#### Participants and Procedure

During the evaluation, we asked the participants to complete a simplified collaborative playlist sharing application. A music microservice and frontend were already prepared beforehand and ready to use. After a short introduction to the

concept, participants were given the task to work on an incomplete album cover microservice model and conclude the scaffolding process with deploying this service. We recruited 14 participants, who were divided into groups of two, with each evaluation session lasting 60 minutes.

After the evaluation, participants were handed out a questionnaire, which used a five-point Likert scale. Most of the users were familiar with REST-APIs and Web services, shown by average responses of 4.64 and 4.57, with Standard Deviations (SDs) of 0.84 and 0.94, respectively. Most were also familiar with the concept of microservices (AVG 4.20, SD 0.91). Regarding the usage of OpenAPI specifications, the participants had very different experience levels, indicated by an average of 3.43 and also confirmed by the high SD of 1.83.

### **Analysis and Outcomes**

The majority of the participants found that the implementation of microservice discovery with API metadata helps them to get a better overview on which services are running and how they could be used (AVG 4.78, SD 0.47). The microservice registry supported participants in deciding which microservice to deploy (AVG 4.43, SD 0.65). Participants also felt that the endpoint details and OpenAPI document in the CAE helped them to be informed about available microservices, with an average of 4.36 and a SD of 0.74. After deploying the microservice, the majority of participants was able to test the API directly in the deployment viewer (AVG of 4.07, SD of 0.92). Participants valued the benefit of having access to this information directly in the browser and not having to actually access the deployment server (AVG of 4.07, SD of 1.07). Regarding the matching process that utilizes the metadata store via the service registry, users found that showing the endpoint information and matching aided them in scaffolding their application, shown by an average of 4.0 and a SD of 0.68.

Overall, the evaluation showed that the microservice discovery system was appreciated by the participants as a first step towards a complete overview on services developed with las2peer (and deployed decentrally in las2peer networks). Nevertheless, the implementation, with a relational database at its core cannot keep the advantages of the decentralized architecture las2peer foresees. Thus, the microservice discovery based on API metadata can only be seen as a first step towards providing (end user) service discovery in decentralized architectures.

## 3.4 Decentralized Service Registry

While the microservice registry based on API metadata provided a way for users to easily explore available services in the network, it was centralized around a database of collected information, stored on single node in the las2peer network, transferring the power over this information solely into the hands of a single node operator. In distributed systems, this type of service registry implementation is not uncommon, as most of them provide a publish-lookup API facilitating service discovery and interoperation. But transferring this concept into the setting of open, decentralized systems is a technical challenge, since the architecture of traditional service registries relies on trusted servers, while existing P2P approaches compromise queriability and security. Beyond this technical challenge, it also raises research questions regarding end user service discovery in the context of online communities. In this section, we introduce the concept of a *Decentralized Service Registry and Discovery Mechanism*<sup>3</sup> by exploiting the tamper-proofness of blockchain technology for our decentralized P2P microservice infrastructure.

### 3.4.1 Motivation

Our use case stemmed mainly from the development of our application and evaluation of building an *Infrastructuring for Knowledge Building* (cf. Sec. 5.2), where we used a las2peer-based setup to support a CoP preparing for a training course of the EVS program. To cope with the diverse background of the participants, the trainers used a form of question-based dialog some days before the actual (on-premise) training course started. This application, consisting of a set of microservices and a Web frontend, enabled users to participate in a sort of mind-mapping process. Our infrastructure allowed members of the community to start a node and all services needed to locally run the application, or only start the node and access services of other members via the network. Another possibility was to just access the Web frontend of another community member to participate. This scenario fulfilled the need for the whole infrastructure being distributed only among the community itself without the need for any central authority.

However, during our evaluations, we identified several shortcomings, once one takes a look at the “bigger picture”. In Fig. 3.9 we depict this scenario. In this example, *Community A* stands for the above mentioned CoP, whilst a

---

<sup>3</sup>The prototype implementation was partially supported by a master thesis [Jans19] under supervision of the author.



second *Community B* also participates in the network. Additionally, we consider a malicious actor Eve. This raises several problems, which we point out next.

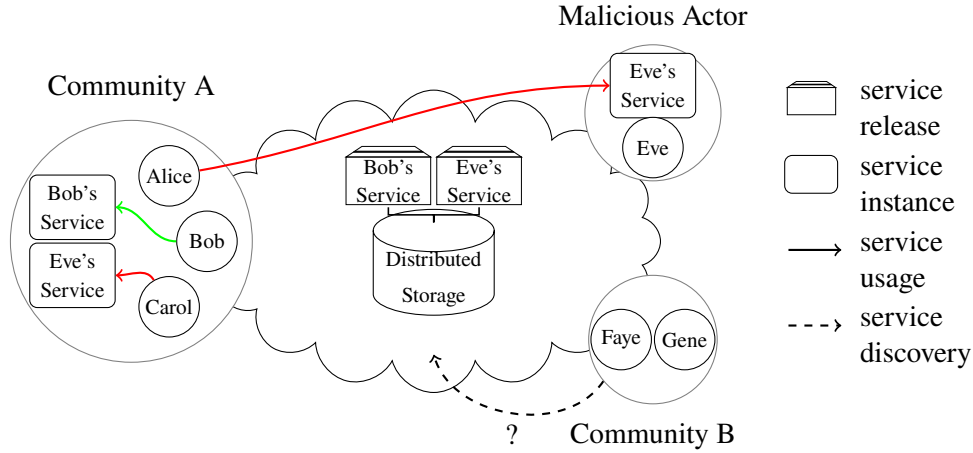


Figure 3.9: Usage scenario without decentralized service registry.

1. *How to explore services available in the network?* When *Community B* joins the network, there neither exists an overview of available service releases nor information on currently deployed service instances in the network. To new communities and community members, the network appears “empty”, and information about services of interest has to come from external sources, like overview websites. While the microservice discovery based on API information introduced in the previous section provides additional service information, this is collected in a centralized way and tightly integrated with las2peers scaffolding environment, which renders it unfeasible for larger, dynamically scaling P2P networks.
2. *Where can I find more information about that service?* The knowledge of services existing in the network might not always be enough to get an impression of what usage possibilities exist. Additional information, like service descriptions, source code location, available frontends, or even usage patterns by other communities may be of relevance to new members or new communities entering the network. Also, the identity of the service developer is of relevance, since trust in a service is highly dependent on its author. In the above example, members of *Community B* might for example be interested in seeing service releases by a particular developer of

*Community A*, e.g., because she is a member of both communities and forms a binding link between them.

3. *How to verify the integrity and origin of a service?* Once a community has established both the knowledge of which service might be worth exploring and where a running instance can be found, P2P networks offer no way of verifying the integrity and origin of services. Specifically, that a remotely running service instance is in fact an unmodified instance of the service release it claims to be. Even when replicating a service locally and checking its integrity via its cryptographic signature, in the absence of a registration authority, the signing key cannot be linked to the (real-life or pseudonymous) identity of the service release's author. This could result in a malicious service instance being executed on the community member's node. In the above example, Bob has published the initial "correct" service release, while Eve publishes a malicious service release that imitates this one. Since there is no way of tracing the origin of a service instance or its release in the network, both communities could accidentally call a malicious service instance. This is depicted by Alice and Carol calling *Eve's Service Instance*, instead of the "correct" one published by Bob.

**Derived Requirements for Decentralized Service Discovery** From the above use case, a number of requirements arise regarding service discovery in decentralized systems. It should enable both end users and developers to easily find service releases, verify their origin and either use remote instances or replicate the release to their own node. Although most of these requirements can be solved by using some kind of central service registry (see also Sec. 2.2), this approach has one major drawback: It redirects the power over the infrastructure from the community to the maintainer of this centralized component and thus contradicts the whole idea of decentralization. Without the knowledge of available services and also the ability to authorize service releases, the community relies on the service registry to forward their discovery requests, which raises the same issues a decentralized infrastructure tries to tackle. To be in line with the concept and preserve its advantages, a decentralized service registry has to be governed by the whole community in terms of authorizing service releases and validating service instances. The blockchain approach fits this idea perfectly.

### 3.4.2 Concept

Our approach proposes a decentralized service registry based on a private blockchain that enables the discovery of services and the secure verification of their release metadata. Combining the completeness and time-preserving properties of a blockchain with space-efficient distributed storage allows us to utilize the strengths of each technology and compensate their respective weaknesses. Specifically, the registry consists of two smart contracts for both services and users. The data written to the blockchain belongs to four types, which are shown along with their respective fields in Fig. 3.10. Because storing data “on-chain” is inefficient and expensive, only essential fields are stored directly on the blockchain, while supplemental fields (marked in italics) are stored “off-chain” in the distributed storage and securely referenced by their hash.

| User Contract   | Service Contract   | Service Release   | Service Announcement  |
|---|--|---|---|
| <b>User Registration</b><br>username<br>agent ID<br>public key<br>Ethereum address<br>timestamp<br><i>email address</i> | <b>Service Registration</b><br>package name<br>author<br>timestamp | package name<br>version<br>timestamp<br><i>title</i><br><i>description</i><br><i>default class</i><br><i>source code repository</i><br><i>frontend endpoint</i> | class name<br>package name<br>version<br>node ID<br>timestamp |

Figure 3.10: The registry smart contract data.

The *User Contract* serves as a decentralized identity management system that ties usernames to their (online) identities via public key cryptography. In contrast to a centralized public key infrastructure, the user has direct control (ownership) of their entry, including the decision to reveal personal data. Thus some users may reveal their real life identities in order to facilitate trust, while others may choose to remain pseudonymous. Registered users can then use the *Service Contract* to publish service releases. This encompasses reserving a service name and linking specific releases to additional metadata. Just like the usernames, these entries are owned by their author by linking them to the author’s public key. Finally, we allow the announcement of service instances, indicating that a user is running a publicly usable instance of the service on their node. Storing this data in a blockchain provides an immutable, auditable historic record of the registry entries

and ensures that they can only be updated by their owners, while also making the data readily available and queryable to all peers.

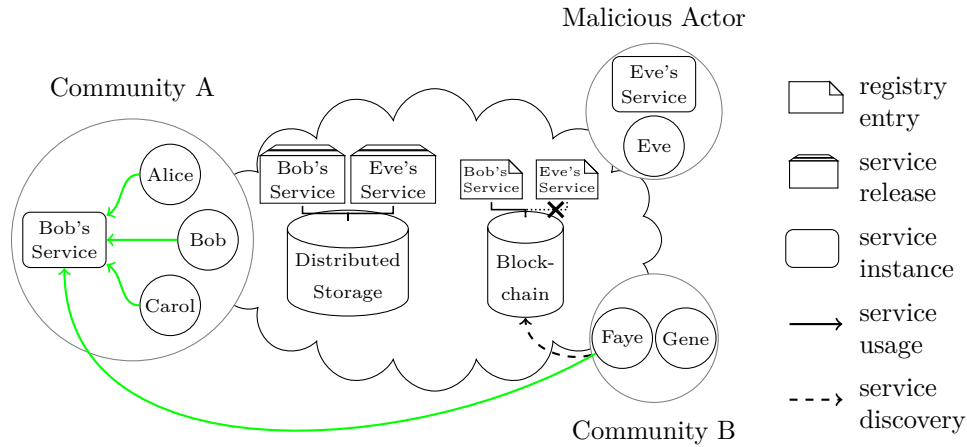


Figure 3.11: Usage scenario with decentralized service registry.

Returning to our motivational usage scenario – with the now blockchain-based service registry – in Fig. 3.11, Bob registered both a username and the name of his service in the network’s decentralized registry. The registry entry for *Bob’s Service* is linked to his username and key pair, which Bob uses to sign his service releases. Eve can still store her malicious, modified release of *Bob’s Service* in the distributed storage. However, she is unable to register it under the same name, nor can she attach Bob’s name to it. All network participants can access the blockchain to see published services and their running instances, and can perform arbitrary queries (e.g., a keyword search over the service metadata). Thus Faye can easily discover *Bob’s Service* even if they are part of disjoint communities. Just like Alice and the other members of *Community A*, Faye also sees the running instance of *Bob’s Service*. If she feels she can trust the user who sent the service announcement and operates the instance, she can access it directly. Otherwise, she can replicate the service locally. By fetching the service release from the distributed storage and comparing its signature against the registry entry, she can verify that the service she starts was in fact authored by Bob.

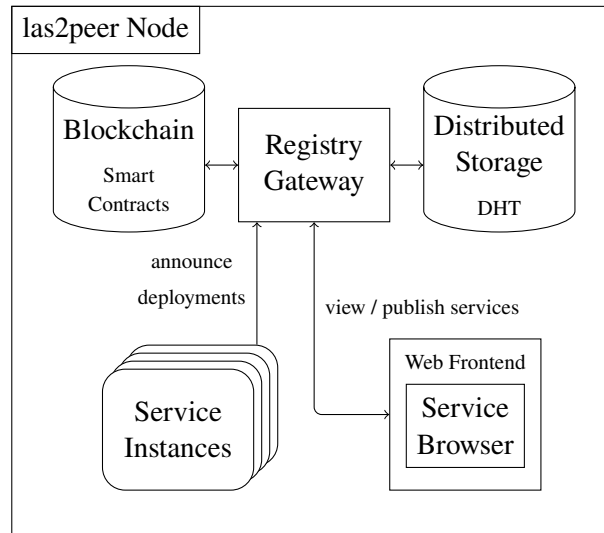


Figure 3.12: Architecture and information flow during common operations.

### 3.4.3 Realization

Fig. 3.12 provides an overview of las2peer’s extended architecture, the new information flow and the integration of the blockchain-based service registry. There are three main components realizing the decentralized service registry:

1. **Smart Contracts:** The foundation of the registry is implemented as Ethereum smart contracts that store and retrieve data from the blockchain. The contracts are written in Ethereum’s high-level scripting language *Solidity*<sup>4</sup>. In addition to the user and service smart contracts described above, a small library contract is employed to handle the verification of signatures for delegated function calls.
2. **Registry Gateway:** Every node contains a registry gateway for accessing the Ethereum blockchain. It transparently stores and fetches the supplemental data fields in the distributed storage, realized as a DHT, and combines them with the data retrieved from the blockchain to utilize the benefits of both storage types. The registry gateway also caches service information to provide efficient lookup.
3. **Service Explorer:** The node’s Web frontend contains a *Service Explorer*

<sup>4</sup><https://solidity.readthedocs.io/>

that allows viewing and uploading service releases, as well as managing local service instances and accessing their frontends. Fig. 3.13 gives an impression of it. This particular example shows two applications, of which one, the “Distributed Noracle” application (see also Sec. 5.2) is only partly deployed in the network (four of six microservices running remotely in the network), while none of the services are deployed on the node the user is accessing. The user can now decide to either start the two remaining services on her node or start all of the services that realize the application locally.

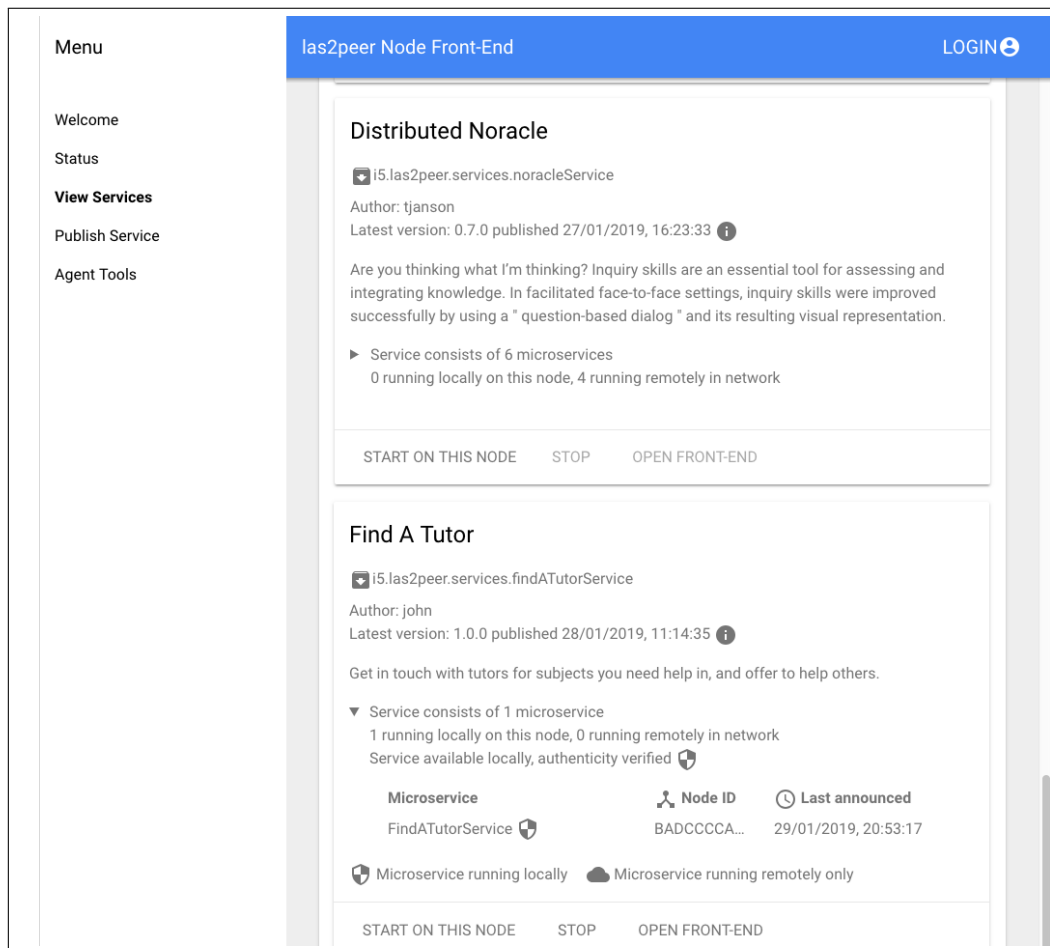


Figure 3.13: Screenshot of the service explorer, depicting two applications currently available in the network.

### Interacting with the Registry

The user and service smart contracts are essentially *name registries* that assign human-friendly names on a first come, first served basis. As such, the user contract provides functions to check the availability, register, and look up the data associated with a username. An important concept of smart contracts is the distinction between *state-changing* functions and those that are “read-only”. The former are processed as transactions appended in a new block (and thus transmitted to and executed by all nodes), while the latter are executed locally and immediately return a value.

**User Registration** When a user wishes to register a username, a read-only function is used to check whether the desired name is still available. If so, we call the registration function, which is state-changing: The call data, consisting of the function name and the arguments, is encoded and broadcast in the form of a transaction signed by the user. When a miner processes the transaction to include it in a block, the arguments are extracted and the call is executed. The smart contract code again checks whether the name is already assigned to someone (e.g., in case that someone else attempted to register the name at nearly the same time). If not, the user entry is created.

We also allow a pattern called delegated function call, in which the user does not sign the transaction herself, but rather prepares a signed certificate of authority (Fig. 3.14). This also contains the call data and is signed by the user. Now any user can prepare a transaction that passes this certificate to a special function of the user contract, which unpacks the arguments, verifies the signature, and registers the username on behalf of the original user. The advantage of this pattern is that it transfers the burden of paying any transaction fees from the user wishing to register to the user who actually sends the transaction. In our use case, we expect that established community members who operate a node may offer to cover the registration fees for users who are unable to run their own node. If the registration was successful, a file containing the supplemental data fields is uploaded to the distributed storage.

**Service Registration, Discovery, and Replication** The service contract operates using the same principles as the user contract. Once again we employ the delegated function call pattern to allow users to cover the transaction fees for a service author. The registration of a service name and publication of a service release is analogous to the user registration procedure. When the deployment of a service instance is

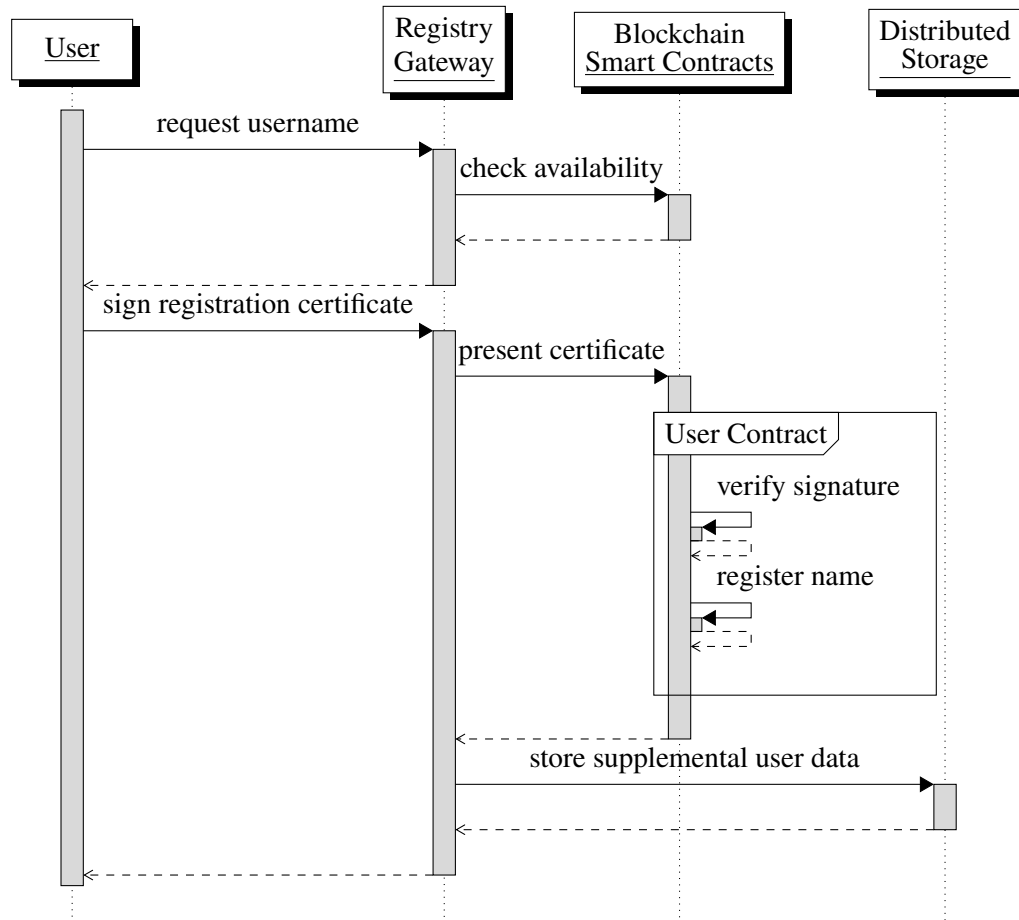


Figure 3.14: Delegated username registration with signed smart contract call.

requested (e.g., through a node's Web frontend), the service release is fetched from the distributed storage. Through two subsequent smart contract calls, first the username of the author of the service release is looked up in the service contract, then her public key is looked up in the user contract, enabling the system to verify the signature of the service release. Once its authenticity is established, the service is started.

### 3.4.4 Evaluation

We carried out a user evaluation to gather feedback for our decentralized user registry. We were especially interested whether users are able to differentiate



between concepts of service ownership and service publishing. Additionally, we wanted to see if users did understand the difference between and services running locally and services running on different nodes than the one that is currently accessed by the user.

### **Participants and Procedure**

We carried out five evaluation sessions with two to three participants each, with a total of 11 participants. The network setup consisted of five permanent nodes and up to three additional nodes started during the session by the participants. Technically, the nodes were started as Docker containers on a single server in order to simulate ideal network conditions. Further, we used a modified version of the Go Ethereum client, `geth`. Since we did not focus on the technical parameters of the blockchain network in this evaluation, we started the client with a very low mining difficulty, leading to short block creation intervals (the “block time”). While many of the participants had experience with software development, the majority was unfamiliar with `las2peer`, our decentralized community service infrastructure. After a brief introduction to `las2peer` and its service registry, the participants were given written tasks that included finding existing services as well as registering a user and publishing their own service. During these tasks, the users first accessed the Web frontend of one of the permanent nodes. Later they accessed a newly started node, that joined the existing network. This hands-on experience lasted about 30 minutes. Afterwards, participants filled out a questionnaire.

### **Analysis and Outcomes**

Fig. 3.15 shows the results of our questionnaire. As one can see, most of the participants were able to understand the basic concepts of the approach and were also able to fulfill the given tasks. We received lowest scores for the question regarding node ownership. This is due to the fact that the majority of participants were not very familiar with P2P infrastructures and mixed up concepts like running services and nodes. Another quite low rating was received for the question about the ease of publishing own services in the network. This was mostly colored by the fact that it was necessary to post exact class names and frontend URLs, that were not checked for correctness in our initial prototype. The remaining questions received quite high scores, which confirms the usability of our developed service registry and its user interface in form of the service explorer. We also asked free-text questions about the relevance of (verified) service authorship. Most of

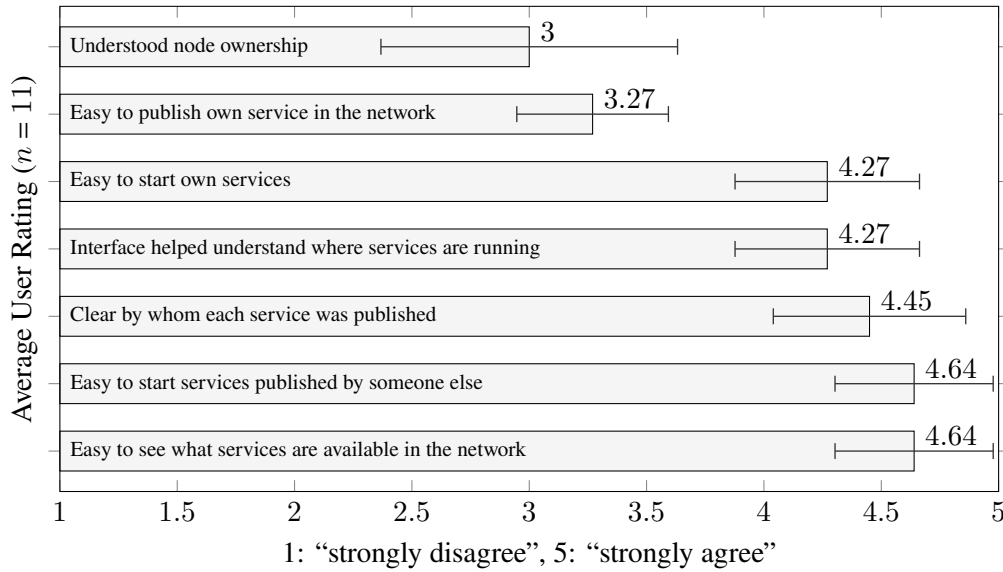


Figure 3.15: Evaluation of the decentralized service registry: questionnaire responses.

the participants stated that this depended highly on the context, and was only relevant for sensitive data. This is in line with our expectations that decentralized infrastructures are valued particularly in the context of sensitive data exchange. We also received interesting statements regarding the question of contribution to the infrastructure, e.g., in terms of computing power. The majority of responses stated that they would support “their” community if resources were sparse and highly in need. Finally, we asked the participants about the trade-off between response time and required storage space. On average, participants were willing to wait up to ten minutes (although a large minority voted for ten seconds or less) for their actions, e.g., service releases and announcements, to be visible on the blockchain.

**Technical Evaluation** A technical aspect that required some additional evaluation was the trade-off between required storage space and block time, as well as between Central Processing Unit (CPU) usage and security, i.e., the integrity of the blockchain data. The block time used in our user evaluation was set to an extremely low value (1 s with a difficulty of 1) that is not feasible for real-world settings, since it requires about 500 MB of space for each node per day, regardless

of the amount of information stored in the blocks. This amount scales inversely with the block time. To study this further, we started modified versions of the Ethereum client with more “realistic” difficulties of  $4 \times 10^5$  and  $128 \times 10^5$ . These numbers are to be interpreted as the average number of hashes required before a new, valid block is found. We ran these clients over a period of a week, letting each mine with a single core of an Intel Xeon E5-4627 v4 CPU, resulting in an average block time of 9.3 s (for the lower difficulty) and 271 s (for the higher difficulty), respectively. Both of these block times are within the acceptable range for the majority of our evaluation participants. Over six days, the blockchain size on disk was 61 MB and 2.1 MB, respectively, which certainly is a more manageable and scalable requirement.

## 3.5 Community Contribution Incentivization

Although improving the security and usage experience of the system, aspects that received little attention in the development of the decentralized service registry presented in the previous section are transaction fees and incentives to participate in the mining process. The blockchain’s currency is specific to its corresponding network and does not have any economic value in itself. Our next iteration of the design science process thus dealt with the question, if it is possible to make the currency represent status in the community. Here, we are both allowing users to provide bounties for certain actions in the network, like the development or deployment of needed services, as well as we are rewarding service developers and hosters for their services within the community, based on the value of their contributions<sup>5</sup>.

### 3.5.1 Motivation

To provide an overview on the challenges blockchain-based decentralized service registries bear, consider the following usage scenario. Alberta and Bryan are part of a CoP that uses las2peer to run its online community services. Alberta has started a las2peer node and has given Bryan access to it, who has developed a new service, which provides a means of securely exchanging messages between peers. Bryan only has access to a relatively weak computer and he knows that Alberta’s computer is stronger and has a faster internet connection, so he deploys the service

---

<sup>5</sup>The prototype implementation was partially supported by a master thesis [Slup20] under supervision of the author.

on the node hosted by Alberta. To increase security and visibility of the service, the binary of the deployment is verified via the blockchain before its instance is launched. Other users join Bryan in using the messaging service and start to share messages via the service daily, as well as regularly sharing large images and other files via the service. While the las2peer infrastructure is capable of automatically scaling up heavily used services throughout all nodes in the network, these services then are not regularly announced on the blockchain (as transaction costs apply) and thereby not discoverable by end users of the system. Thus, as much as Bryan's contribution in developing, preparing and registering the service was valuable, it is now up to Alberta, to not only provide computational resources for the running service, but also to pay the transaction fees for announcing the service deployment along with its verification value to the blockchain. This leads to the problem of *duplicate resource spending* for users who are contributing infrastructure to the community.

Reimbursing service hosters and developers for their efforts and costs provides a way of alleviating this issue. Therefore, we concentrate on a remedy for the duplicate resource spending issue of hosters, while also creating an incentive for service authors to continue developing services for their communities. A user reputation system creates an additional layer of motivation to provide and develop high quality content. Constraining the cryptocurrency and reputation to the decentralized infrastructure network is in line with the assumption that online reputation systems should by design be context-specific to remain relevant.

### 3.5.2 Concept

This section describes our approach how to incentivize contribution in a decentralized architecture. We describe the improvements our contribution delivers to the usage scenario we described earlier, how the developed reputation system uses cryptocurrency to incentivize community members, and how the payout is calculated.

#### Community Currency - L2Pcoin

Revisiting the usage scenario, to motivate Alberta to remain an active member of the community and incentivize others to follow her and provide their share of computational resources to the community's infrastructure, a system based on reciprocity should reward users for their contributions to the community. Both Bryan and Alberta, as developer and hoster, should be entitled to a fair share of

cryptocurrency in return for their input. To prevent users from abusing the payback system and also to provide a direct way of user feedback, a karma rating system is integrated in the calculation of the reward.

Since the amount of reputation, which is paid out to hosters and service developers, should not only be based on the karma of the requester, but also on the value their services are providing to the community, we renamed the Ether cryptocurrency in the context of las2peer and use the acronym *L2Pcoin* instead. While the application of a reputation system is to directly incentivize users for their contribution to the community, the paid out cryptocurrency requires a backing in the community to actually provide value to the users. The simplest form of such backing is the restriction of certain actions in the community to users who have not yet accumulated a certain amount of reputation, i.e. introducing a ranking system. However, one major argument against the introduction of such a system is the exclusion of new users from key parts of the system. Instead, the concept of a *bounty* or *reward* is used as a more advanced mechanism to provide backing for *L2Pcoin*. Here, users can receive or send direct rewards either in response to, or in anticipation of, actions in the community. Service developers could automate and categorize such rewards in their applications and specifically reward community users for their contributions in form of, for example, popular blog posts or insightful wiki entries. Sending developers *L2Pcoin* can be used to motivate them to fix bugs or extend the functionality of their services.

### Calculating the Payout

To calculate the payout for each participant of the network, we focused on three fundamental factors of the decentralized service provisioning network: *Service Development Value (SDV)*, *Service Hosting Value (SHV)*, and *User Rating (UR)*. Their range is a floating point value of 0, 5, mapped to a 5-star Likert-scale rating. While the values needed to calculate the *UR* are provided through karma voting via a node's Web frontend (see also Sec. 3.5.3) and stored on the blockchain, *SDV* and *SHV* have to be specified and agreed on by the community. Here, we make use of las2peer's *NRT Evaluation Center* (see also Sec. 4.3.4), and specify two success factors called *Service Hosting* and *Service Development*, which the community has to integrate into their corresponding service success model. Our reputation system queries these to calculate *SDV* and *SHV*, respectively.

To remedy restricting voting power to members of the community, the reputation payout value is multiplied by the global user rating of the requesting agent. Both the author of the service and the hoster of a node are advertised in the Web

frontend of a node, so users can directly see the rating of authors and developers to establish a trusted interaction with the service. In summary, the reputation payout formula *for each service* a user has either authored or is currently hosting on her node is obtained as follows:

$$u * UR * d * n * SDV + h * m * SHV$$

**UR** is the average user rating, i.e. the ratings received by the user, divided by the amount of votes received. This value is set to 1 if the user does not have a registered reputation profile (i.e. cannot have received any votes) to allow fresh users to be paid out *L2Pcoin* for service hosting and development before they have the funds to pay for the transaction required to register a reputation profile.

**SDV** is the service development value of the service.

**SHV** is the service hosting value of the service.

**n, m** are the amount of (blockchain) announcements for this particular service since the last payout request. These values are set to 0, if the requesting user is not the author or hoster of this service, respectively.

**u, d, h** are weights which can be used to influence the impact of *UR*, *SDV* and *SHV*. These values are fixed at the first node's startup and can be used to fine-tune the calculations according to specific community needs. They are deliberately not modifiable at runtime to provide a consistent reputation payout to increase user trust in the platform.

### 3.5.3 Realization

This section describes the major parts of the implementation, which consists of a reputation system and a payout mechanism to reimburse service hosters and developers for their community contribution.

Fig. 3.16 provides an overview on the functionality. A user visiting a las2peer node frontend can login or register a new account and thus becomes a registered user of the community. This automatically creates an *Ethereum Wallet* and assigns it to the user profile. Now, the user can participate in the reputation system by choosing to opt-in. This allows her to see a list of other users in the network and provides the ability to rate them. Sending cryptocurrency (rewarding others) is

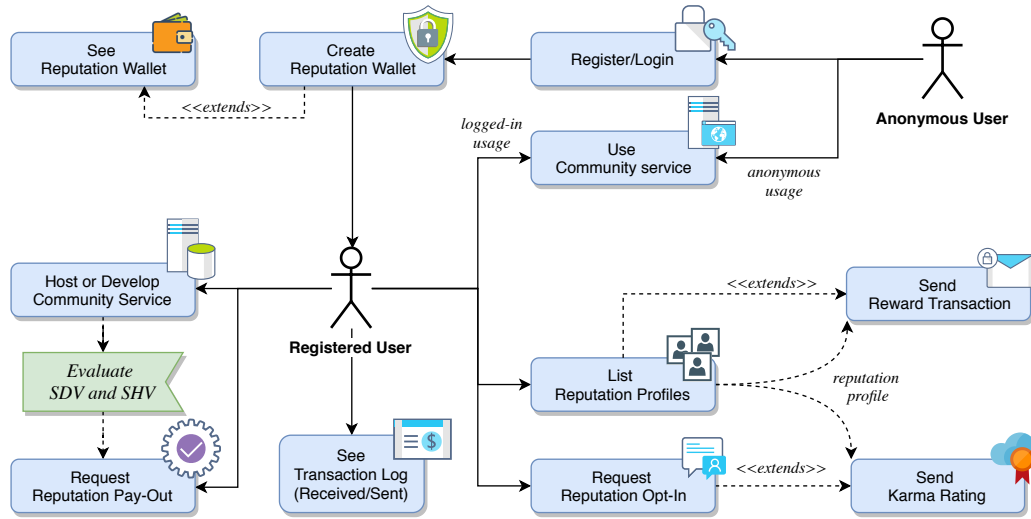


Figure 3.16: System overview of the incentivization system.

not restricted to the opt-in reputation system, so users can do that regardless of the opt-in status of the sender or recipient. Users can become service developers by authoring a Web service and uploading the associated service binary through the node’s Web frontend. Becoming a service hoster requires running a las2peer node, which has the user email address that is provided at startup linked to the corresponding Ethereum wallet. Once a service has been started on a node, it begins emitting its announcements to the blockchain. Additionally, registered users can see the history of their interactions with peers and the system via the transaction log.

### Calculating Service Uptime

As the reputation payout is tied to a user action (i.e. a button click), this could be abused by triggering the button multiple times. A time-lock mechanism seems an intuitive solution, but relying on a time-based solution to prevent abuse of the payout system would require all nodes in the network to have a common time source. Clock synchronization in networks is a well known problem in computer science [LLSW10], yet its solution is non-trivial, especially in large and decentralized networks.

To remedy this, we chose a different approach and use the *amount of service announcements* since the last payout. As services get periodically announced

on the blockchain, we can base our calculations on this frequency, as even if a service would be announced too frequently or rarely, this is easily detected, as all announcements are public on the blockchain. Since time-based calculations on the blockchain are not exact, determining the payout time window has been indirectly based on the number of blocks that have been mined since the last request.

A Red-Black tree [Hank99] was employed to provide a three-layered data structure. The topmost layer (which gets used in the sorting of the tree) is the increasing block number. In the second layer, a list of services which were announced during this block is saved. The leaf set is comprised of a collection of nodes hosting the services of the second layer. This data structure has been chosen due to the ability to filter the first layer in such a way, that only those blocks that have a larger block number than the provided parameter are returned. The ability to partition the first layer of the transaction log tree to query only values larger than a certain threshold (called the *TailMap*) is used to only process blocks which are relevant to the payout time window.

### The Reputation Registry

To realize the reputation functionalities, we introduce the *Reputation Registry*, a smart contract. It enables user voting and keeps track of reward transaction metadata. The following events occur in the life-cycle of the smart contract:

**User Profile Created** This event carries information about the mapping between the Ethereum agent and the relevant reputation profile in the smart contract. Successfully registering a profile triggers this event.

**Transaction Score Changed** This is used to listen for changes in the score of a reputation profile. This event gets triggered every time a user's reputation changes and includes information about the vote sender, recipient and the new score of the profile.

**Transaction Count Changed** This is used to listen for changes in the transaction count of a reputation profile. This event gets triggered whenever the number of transactions sent or received by a user has changed.

**Transaction Added** Every time a user rating transaction is sent, this event gets triggered to notify about the sender / recipient pair, to indicate the transaction timestamp and to show the grade and new total score of the recipient.



**Generic Transaction Added** This is needed to implement the reputation reward mechanism that allows users to attach metadata to a *L2Pcoin* transfer. The event gets triggered on every generic transaction in the system and includes a sender / recipient pair, the timestamp of the transaction, a message, its type and the transferred amount with its transaction hash.

### Reputation Dashboard

To introduce the frontend for the community contribution incentivization system, las2peer node’s frontend was extended with a *Reputation Dashboard*. The landing page of the dashboard is depicted in Fig. 3.17. Participating users can vote others or send them rewards in this part of the UI. The user list shows the count of sent and received votes of the user, along with their average score. Users can hover over this score to cast their votes. The “Actions” button provides the ability to directly send *L2Pcoin* to this user. A tabbed interface provides access to incoming and outgoing voting history and the list of all previous reputation payout values.

| Username | Rate user | Reputation Statistics  | Wallet address                             | Actions |
|----------|-----------|------------------------|--|---------|
| Carl     | 👍👍👍👍👍     | 👍 (4.50)   🗳️ 2   🗳️ 2 | 0xd8678ab2d06a79a1e6d494322ff8509711774152 | 👛       |
| Francis  | 👍👍👍👍👍     | 👍 (4.00)   🗳️ 1   🗳️ 0 | 0x4f59490eaf19e225b4cac8bc9b96b592c9135c89 | 👛       |
| Boris    | 👍👍👍👍👍     | 👍 (1.83)   🗳️ 6   🗳️ 0 | 0x8426c6aa3464e3b05c060dd7ff378f37ff4ebffb | 👛       |
| Daniel   | 👍👍👍👍👍     | 👍 (1.00)   🗳️ 1   🗳️ 1 | 0xf8e8a0c8132cd9f8d91b6836883a8233abd403c4 | 👛       |

Figure 3.17: Reputation dashboard landing page, showing a list of community members together with their reputation.

To allow users to differentiate how much trust they have in the author and hoster of a service, we extended the service explorer’s UI to encompass the ratings of both service author and hoster (see Fig. 3.18). The reputation wallet view shows a user’s *L2Pcoin* balance along with her user rating score (see Fig. 3.19). Here, users can request the reimbursement for their community contribution to be paid out and choose which community to have their services rated by. Additionally, users can see the balance of the coinbase account, which is used for the reputation payout and represents the total amount of *L2Pcoin* that can be transferred by means of this mechanism. The opt-in button (not depicted in the screenshot because the user has already opted-in) of the reputation system can be found underneath the reputation payout button. Still, users can request a payout without

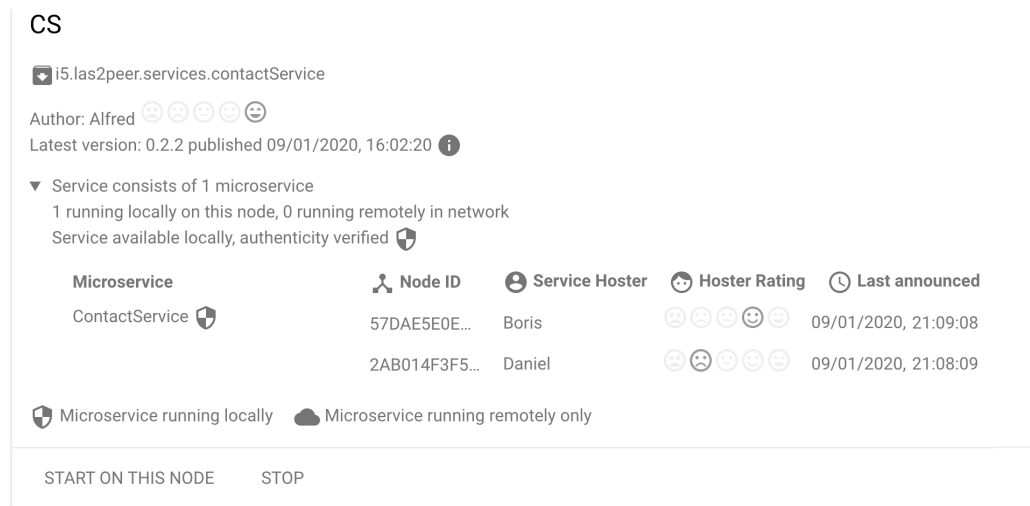


Figure 3.18: Service author and hoster reputation, integrated into the service explorer of the decentralized service registry.

being opted-in to the reputation system. This decision has been made, as opt-in (i.e. registering a reputation profile) requires cryptocurrency for its transaction fees. Without existing accounts transferring funds to a newly registered profile, acquiring enough currency to pay these fees (through the system UI) can currently only be acquired by requesting a reputation payout. To welcome new users, this payout defaults to a small amount of *L2Pcoin* on its first trigger, that will allow the users to opt-in and participate in a few transactions to generate more. When

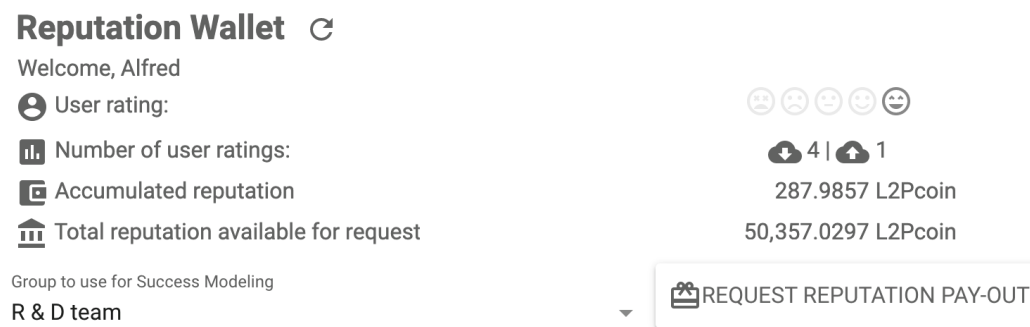



Figure 3.19: Reputation wallet view.

a user requests her reputation to be paid out, the system calculates the amount of contribution the requesting user has provided (see Fig. 3.20). The time-window for


this calculation is the time since the last payout, or since the start of the blockchain, whichever is more recent. The payout formula, together with the user's (hosted and/or developed) services, as well as the individual scores for each value taken into consideration while calculation the payout, are presented transparently to the user to further increase the trust in the system.

### Reputation Pay-out Log - transaction successful.

 **UserRating Score :** 3,

 **HostingServices Score:** 20 , rewarded for the following services:

- i5.las2peer.services.contactService.ContactService: 4 announcements with a rating of 5

 **DevelopServices Score:** 0. No developed services detected.

 **Total Reputation Payout:** 60.0 L2PCoin

The reputation pay-out has been obtained as follows:

$$\frac{\text{UserRating} * 100\%}{\times [ ( \text{HostingServices} * 100\% ) + ( \text{DevelopServices} * 100\% ) ]}$$

Figure 3.20: Reputation payout confirmation dialog.

### 3.5.4 Evaluation

We conducted a user evaluation to test the applicability of our community incentivization system. The main goal was to gather feedback regarding the usability of our approach from an end user perspective, as well as to validate, that the approach is understood and users see the proposed value of it.

#### Participants and Procedure

Our evaluation involved 14 sessions with one participant each. Participants were recruited from the faculty for computer science, both students and researchers. A hands-on session of about 30 minutes was followed by a questionnaire the participants had to fill in. The questionnaire consisted of questions to be answered on a 5-point Likert scale. The participants were first given a short tour of the system's functionality. After that, the participants were asked to perform tasks like to review services running in the network, opt-in to the reputation system,

register a new (already prepared) service and upload it to the network, rate users and observe received ratings (fabricated by the evaluation facilitator).

### Analysis and Outcomes

The results (Fig. 3.21) show that most users understood the interface and the functionality it represents. We received the lowest responses on the question

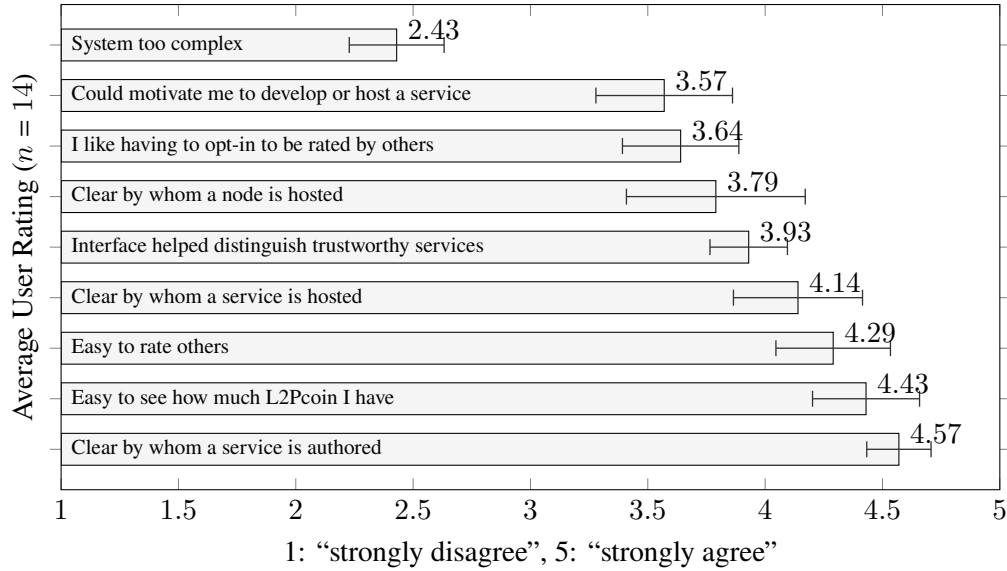


Figure 3.21: Evaluation of the community incentivization system: questionnaire responses.

of reputation complexity, however the answers indicate a double-negative (participants disagree that it is too complex, meaning they indicate that it is not over-complicated), so overall the result is positive. Other lower scores have been awarded to the questions of *motivation to contribute by hosting or developing a service* and the *opt-in to the reputation system*. A low score on the question whether users feel motivated by this system to develop or host a service can be attributed to the fact that service developers did not feel that their reward was adequate for their time and expertise necessary to develop a service, as they did not see value in the reimbursement. In other words, participants often quoted the lack of things to spend *L2Pcoin* on. This is mainly due to the artificial setting of our evaluation and we will have to meet this challenge in future work on this topic. The opt-in mechanism to the reputation has been met with mixed responses, as some users

expressed their concern, that users who do not participate in the system are most likely those who deserve a rating the most, and usually it is a low one. However, with average scores of 3.57 and 3.64, the responses to these questions are still high enough to lead us to the conclusion, that the first reactions to our proposed approach are positive.

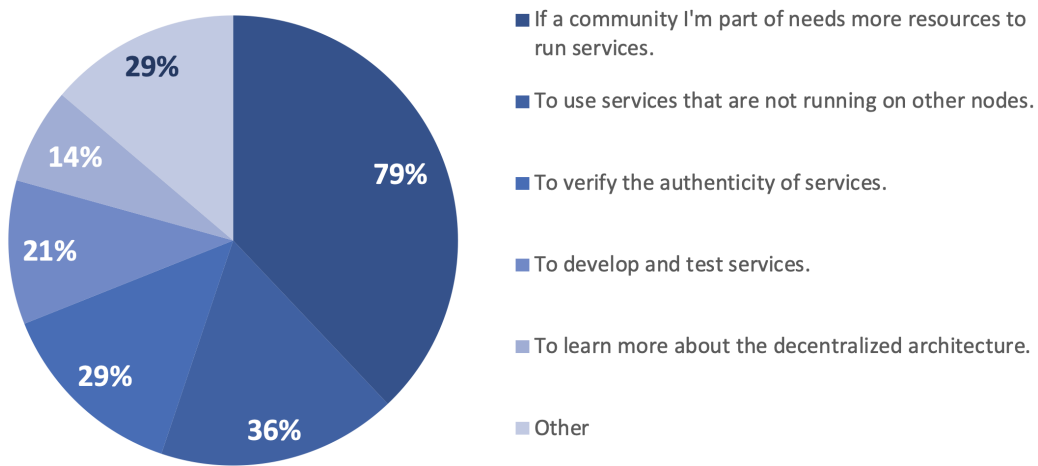


Figure 3.22: Evaluation of the community incentivization system: For which reasons would you consider hosting a service for a community?

To gain a better understanding of the reasons for community service hosting, a multiple-choice question was posed at the end of the evaluation (see Fig. 3.22). The vast majority of participants declared they would be inclined to contribute their computational resources to their community, if there was a deficit preventing the efficient functioning of the community. 36% of the participants would provide resources in form of a participating node to host the requested service if it is not running on any other nodes in the network. Slightly short of a third (29% each) would host services to either verify their authenticity, or to develop and test them. Other options included gaining *L2Pcoin*, obtaining feedback about self-developed services, learning more about the decentralized architecture and making sure that the service in question is really available.

## 3.6 Blockchain-Based Verification and Consent Management of Learning Analytics Data

Our final contribution in this chapter extends the blockchain infrastructure developed in the course of the last two sections (and design science iterations) to verify collected LA data and provide blockchain-based consent management. Applied in a large scale mentoring scenario (see also Sec. 5.3), we here introduce the infrastructure of this application<sup>6</sup>.

### 3.6.1 Motivation

Despite being a vital component of educational processes, mentoring relationships in today's higher education are hard to maintain. This is caused by the mismatch of the number of mentors in relation to the number of mentees enrolled in higher education programs. Consequently, this creates the need for scalable technical support for automated mentoring processes. In addition to the right tools, LA that require students' personal data, make mentoring effective. This LA data is sensitive, as it reflects the learning behavior, which involves personal information such as grades. Consequently, the consent of each individual learner has to be requested prior to the extraction and analysis of their personal data. Security and transparency measures are required for collection and processing of the data, to fulfill data processing regulations such as the European GDPR<sup>7</sup>.

In order to show which issues arise with these settings, we consider a typical LA data collection situation as shown in Fig. 3.23. Our use case stems from the large application and evaluation within a scalable mentoring architecture described in Sec. 5.3 and assumes the there given infrastructure, without limiting the here depicted issues to this particular scenario. For the scope of this example, we consider LA data stemming only from LMS, although we acknowledge and support the collection from multimodal sources in a very similar way. Additionally, we assume the mentoring process being conducted by a so-called intelligent mentoring bot (cf. Sec. 4.6) that provides the learners with feedback, based on the collected LA data. Learners are informed that behavioral data, generated in the LMS they use in their class, will be analyzed to provide personal mentoring. Prior to the beginning of the course (and thus the mentoring process), the students

---

<sup>6</sup>The prototype implementation was partially supported by a bachelor thesis [Beng20] under supervision of the author.

<sup>7</sup><https://gdpr.eu/>

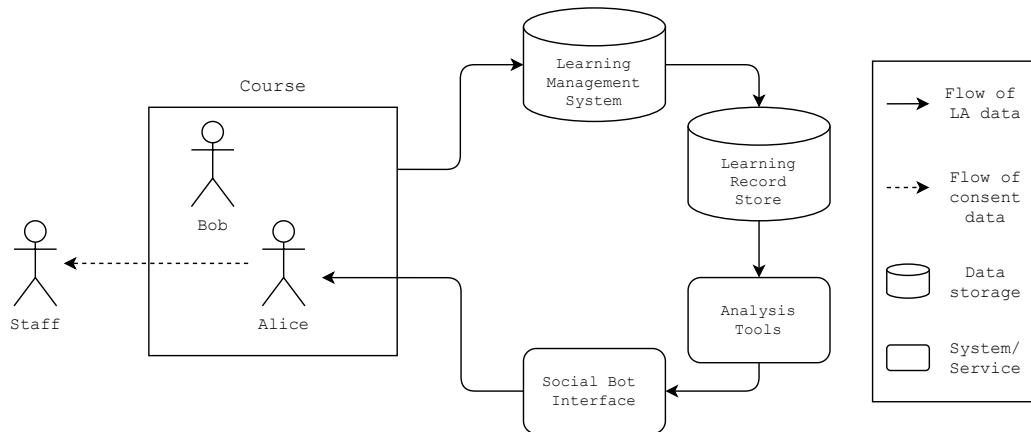


Figure 3.23: Usage scenario without verified LA data.

are asked for their consent to the processing and analysis of their data. The student Alice consents to the collection and use of her data while the student Bob does not consent (e.g., due to personal privacy concerns). Their consent is manually managed by members of the mentoring team (or within a separate system). The students then proceed to interact with the course material in the LMS. As the course progresses, Alice receives personalized feedback based on the data provided. The recommendations are generated based on the analysis of data that has been transmitted from the LMS to, in our example, a LRS. Assuming that Alice does not agree with the feedback or is curious as to what the basis for the feedback is, she wishes to access the stored data or revoke her consent. Based on this usage scenario, several concerns and questions arise in the collection and processing of the learners' LA data.

1. *How to better specify and handle student consent?* While the current process can generally be aligned with the principles for privacy in LA, it requires manual work on the side of the mentoring team. Furthermore, if not integrated into the data extraction process itself, it is even harder (although possible) to differentiate which type of personal data the learner agrees for the collection and processing. This usually implies that there is only one uniform consent collected from all learners within a mentoring process. A limitation of consent to specific data - for instance to grades and submissions only - is usually not done or requires substantial manual effort. Moreover, if a learner revokes her consent after the beginning of the course, this can again only be managed with manual effort.

2. *How to verify, that only consent-given LA data is persisted?* Personalized mentoring recommendations, as well as the general implications of the analysis carry weight for students and educational institutions alike. As a consequence, depending on the learners' trust in these recommendation, manipulation of the underlying data in the LRS could harm the students' learning process and results. While giving learners the ability to access the stored personal data is surely beneficial, it can not be expected that learners themselves verify whether all data contained is correct.

Ideally, solving these two issues also increases the learners' trust in and acceptance of automated mentoring by keeping their data in their hands, confidential and secure. In the following, we tackle the issue of enforcing the learners' preferences on data processing by using a blockchain-based consent management and verification of LA data. Thereby, we maintain a record of the learners' accessed personal data and record their consent to the collection and processing of LA data.

### 3.6.2 Concept

On a conceptual level, the approach consists of three main parts. The first two are extensions of las2peer's accompanying blockchain, namely the verification of LA data and the consent management. The third is the chatbot as conversational interface for learners to access their LA data logs and manage their consent, which we developed using the SBF (described in detail in Sec.4.6). It has to be mentioned here, that this bot must not necessarily be tied to the actual mentoring process itself (although it could be the same bot), but can exist on its own, with the sole responsibility of managing the consent and providing an insight into stored LA data.

To achieve the verification, we use the blockchain-based registry that stores references to all LA data extracted from the LMS. This is vital for the verification, as the creation of immutable references on the blockchain enables the detection of changes in the data that would go unnoticed otherwise. This process of storing a reference and comparing against that reference later is a widely established use-case for blockchain technology. By storing hashes for reference only and relying on the irreversible nature of hash-functions, we discourage access to personal data on the blockchain. Thus, although blockchain data is immutably, it only stores references and once the data itself is deleted, no personal data of a learner remains in the system.



For consent management, we provide learners with an interface that allows direct interaction with the system. Issued consent is stored on the blockchain and is integrated into the extraction process to restrict access to personal data, for which no consent was given. This is done by querying learners' consent before their data is extracted from the LMS. Our concept includes a chatbot for students to use the features for consent management and verification of LA data. This bot enables the storage and revocation of students' consent, the display of their currently supplied consent and a listing of the collected their personal data.

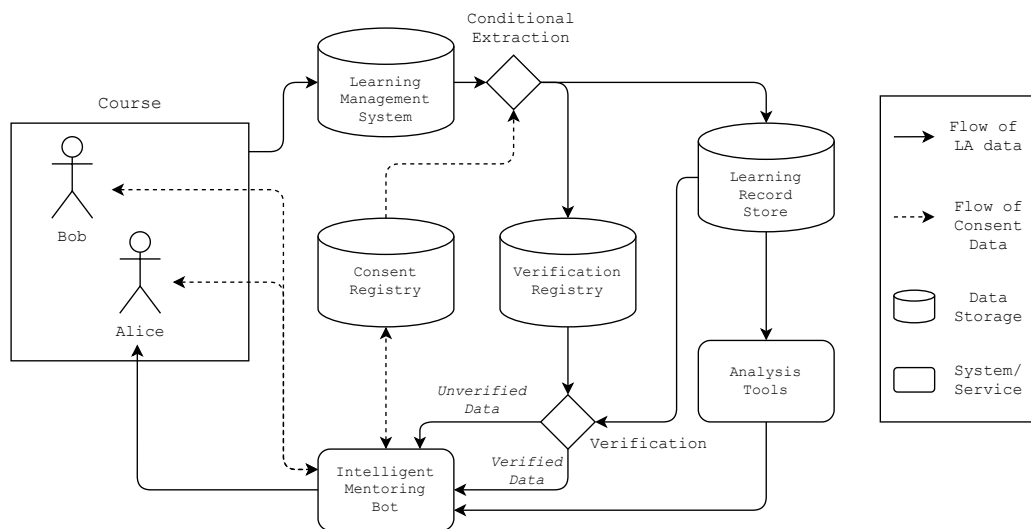


Figure 3.24: Usage scenario with verified LA data.

Revisiting the usage scenario described above in Fig. 3.24, we can see that these new processes and features help to cope with the raised issues. First of all, we eliminate the need for manual interference of staff members for consent handling. At the beginning of the course, Alice and Bob are both informed that their consent is required before they can use the personalized LA features. Alice now stores her consent to the collection of her personal learning data from the LMS with the help of the mentoring bot interface, while Bob does not give his consent. However, they then proceed to use the same learning course room. Before data is extracted, Alice's and Bob's consent is queried from the *Consent Registry*. With only Alice having given her consent, only her data is extracted and stored in the LRS. Alice and Bob both can ask the bot to display their collected personal data from the LRS at any time. The bot then provides this data and based on a comparison with the reference in the verification registry, it is indicated whether

the data can be traced back to an extraction from a source respecting the learner's consent (verified data) or not (unverified data). Based on a comparison with the reference in the verification registry, it is indicated to her whether the data can be traced back to an extraction from a valid source – in this case the LMS course room.

### 3.6.3 Realization

Our realization is based on a las2peer network with four services, a bot running within it and MobSOS used as a forwarding mechanism for the LA data. An overview of the system can be seen in Fig. 3.25. A las2peer *Data Proxy* service is responsible for extracting data from a LA data source, by using the API it provides. This process is very platform specific. As an exemplary data proxy, the *Moodle Data Proxy* accesses its LMS by periodically calling the RESTful API of Moodle for data extraction. The data proxy service then transfers the raw, system specific data, into xAPI statements. It has to be noted, that the implementation supports the extraction (and aggregation) of LA data from multimodal sources, thus there can be multiple data proxy services, which can extract data from different sources. The direct transformation into a unified data format allows for the LA data to be treated equally from this point on, regardless of its origin. As these data proxy services often reside at the premise of the data source, further sending of the data to the LRS is done via the las2peer end-to-end encrypted communication system, to allow for secure data transportation. Here, we make use of MobSOS to send data from the proxy services to the *Learning Record Store Proxy* service. This service analyses the data and provides and unifies the learners' data from multimodal sources to provide a unified view on each learner. From here on, the data can be analyzed and used for automated mentoring.

The process depicted up to this point does not yet take the verification and consent management into account and could be seen as a realization of the scenario depicted in Fig. 3.23. As the central point of managing both consent and verification of LA data, the *LA Verification* service can store, show and revoke consent, and it is responsible to check learners' consent whenever services request to collect or process data. Furthermore, the service handles the verification of stored LA data and the display to the user.

For storage and retrieval of learners' consent, a smart contract, the *Consent Registry*, is used. Using a social bot, learners give their consent, based on pre-configured consent options and the LA verification service stores this consent on the blockchain. Whenever LA data is extracted from a data source, the learners'

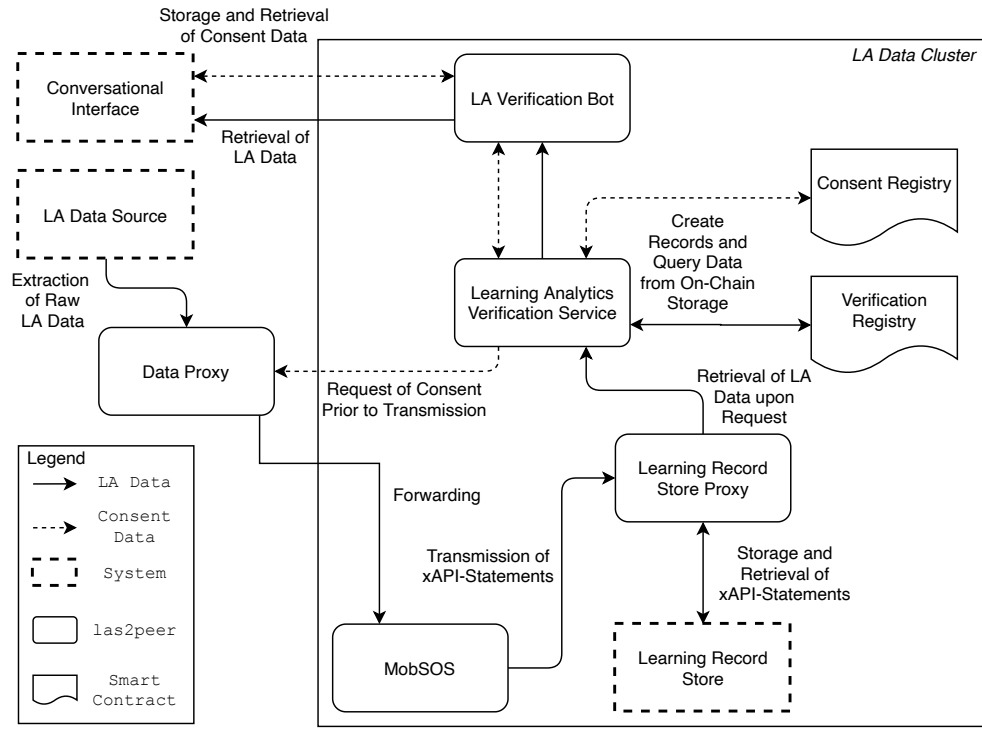


Figure 3.25: Architectural overview of the LA data verification system.

consent is checked by a call of the data proxy service before the data is transmitted. If the consent is given, the previously described process of transmission through las2peer using the MobSOS pipeline and the LRS proxy service is carried out, and the data is stored in the LRS. In addition, a hash is created in the LA verification service and this hash is stored in the *Verification Registry* on the blockchain. The hash is generated based on a string that contains the actor (identified by an email-address), the action (verb of the xAPI statement), the object used in the interaction, and a timestamp. When students request access to their stored LA data, the LA verification service calls the LRS proxy service. The obtained data is hashed again and it is checked if this hash has previously been recorded in the verification registry.

Both the verification registry and the consent registry follow a similar approach as the existing smart contracts in las2peer. Only essential data such as hashes and timestamps are recorded on the blockchain, while other data is stored off-chain. In the case of the service registry, data is stored in the shared storage of the las2peer network, while in the case of the verification registry, the LA data is stored in the

LRS in the form of xAPI statements.

One of the key characteristics of using blockchain is the immutability of data. This is a challenge for the revocation of consent. While it is advantageous for learners to know that the consent is securely stored on the blockchain and cannot be changed, learners still have to be able to revoke their consent. However, the main requirement in this case is that services do not access the previously stored consent anymore. This is achieved by using the a revoke consent function in the consent registry to store an empty consent for the given user. While this does not delete the previous consent entirely, as the data would still be available in the transactions stored on the blockchain, it effectively voids the consent from the perspective of the LA verification service. Consequently, each subsequent query for consent of the learner will trigger a rejection as no consent levels are stored for the learner.

### **User Interface**

The LA verification bot serves as the interface for the student to interact with the implemented features for consent management and verification. Based on pre-defined intents, the bot handles user commands to navigate to both the menu for consent management as well as the display functionality for verified LA data. In addition to the main functionality, the bot displays introductory information for students that explains the basic concepts and implications of the given consent and verification of their data.

Communication with the bot is expected to be rather infrequent. After all, the consent will be given once initially and not revoked or updated often. Due to this, the menu is designed to allow precise execution of given commands and uses short text- and number-based input for choosing from different options in the consent menu. At the same time, it is still necessary, to supply information for learners to understand the services that are available to them. Fig. 3.26 shows an example of a conversation between learner and LA verification bot for the storage of the learner's consent. Due to the nature of the application the bot was evaluated in, it was developed with a German language body.

### **3.6.4 Evaluation**

We evaluated the approach in a small user evaluation, with a focus on two aspects. On the one hand, we aimed to evaluate the usability of the overall system. On the

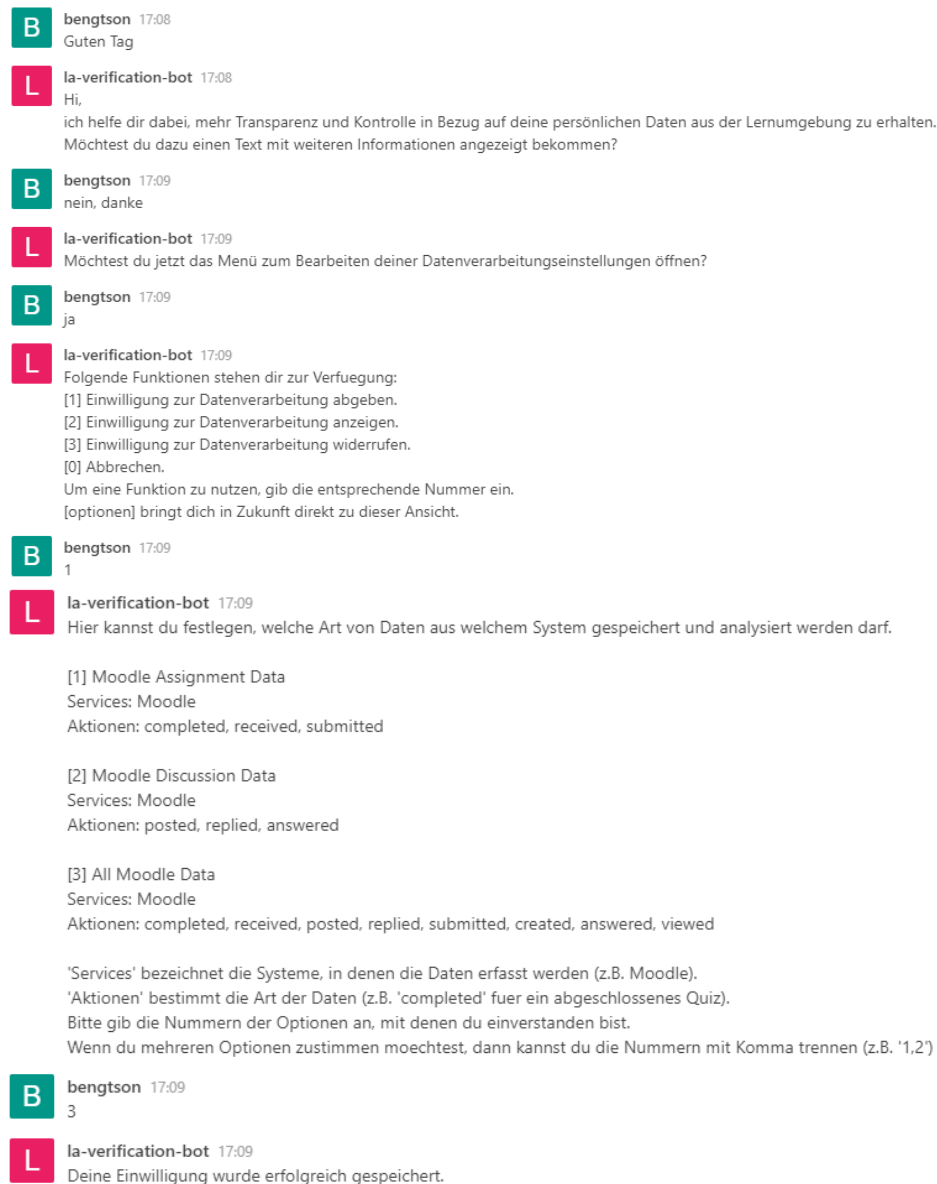


Figure 3.26: Sample conversation with the LA verification bot.

other hand, we were interested in how the features effected the learners' perceived control over and security of their data.

### Participants and Procedure

We evaluated our approach with both mentees (students) and mentors (teaching staff). Altogether, we recruited 10 students and 6 faculty members of multiple universities. During the evaluation, first a usage scenario without the blockchain-based verification was presented (roughly as sketched in Fig. 3.23), to show the motivation behind the added features. Then, the verification of LA data using blockchain technology was explained. After the introduction, participants received a list of tasks for them to execute while screen sharing was active, so that the individual steps could be observed. The participants went through the complete process as depicted in our exemplary usage scenario in Fig. 3.24. After the evaluation, participants were handed out a questionnaire, which used a five-point Likert scale. All evaluations were carried out online.

### Analysis and Outcomes

Our first set of question was targeted towards the general opinion regarding personal data collection by Web services. Even though most of the participants were skeptical about access to their personal data (AVG of 3.56, SD of 1.17), only half of the participants expressed confidence in their ability to make informed decisions about giving their consent to the collection and processing of their personal data (AVG=3.00, SD of 1.00). Interestingly, the group of mentees felt more confident in their ability to make informed decisions with a median of 3.5 compared to a median of 2 for mentors. Results also showed that the right to request access to the stored personal data from institutions or companies is known, but only about half of the participants stated that they have previously used that right (9 out of 16). Interestingly, some participants that agreed to being skeptical, also answered that they did not make informed decisions and that they did not request their stored personal data before. We interpret this discrepancy in the attitude towards collection of personal data and the ability to make informed decisions, as well as the willingness to take control over personal data, as a need for transparency on the side of services that process personal data.

Fig. 3.27 shows selected results of our questionnaire. Here we can see, that most participants understood the process of verification and the implications of the provided information on their verified data, with a slightly higher agreement in the group of mentees. Among the comments received on the verification, one participant suggested to supply additional information on the blockchain in general, as well as the verification process through the chat bot on mentees' request. This

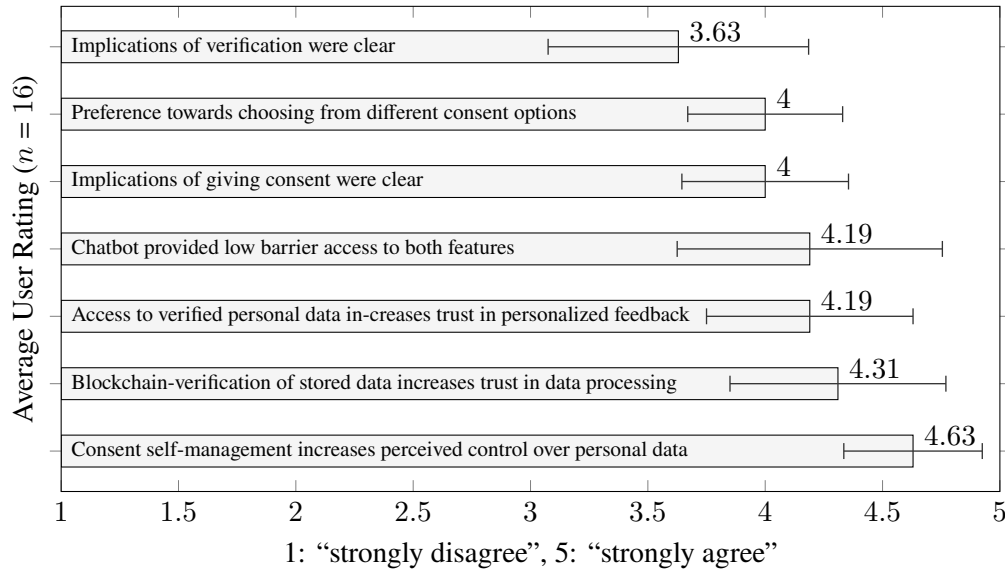


Figure 3.27: Evaluation of the verification system: questionnaire responses.

is backed up with the observations from the evaluation sessions, where several questions were directed at the nature of the process and meaning of the verification in particular. Another interesting observation was, that some participants that initially disagreed to have a basic understanding of blockchain agreed that the presented verification using the blockchain increases their trust.

In both groups of stakeholders, the participants agreed to the statement that they preferred the more fine granular consent management over the decision to just consent or not consent, with slightly stronger agreement in the group of mentees. Two mentees explicitly mentioned that they liked the idea of being able to restrict the extend of data that is recorded in the system.

The evaluation showed the applicability of our approach and both mentees and mentors valued the idea of having verifiable consent-management and LA data processing. The chatbot provided low barrier access to both features. Even if only applied in a simulated mentoring environment, participants found the consent self-management functionalities increasing their perceived control of what LA data was recorded and stated that the trust towards the secure handling of their LA data was increased due to the verification features.

The results from the questionnaire and the observations from the evaluation sessions indicate, that there is an interest in the provided functionality of consent management and verification and that this can contribute towards improving the

transparency and control of students over their data within the scope of LA projects. While the sample size in the evaluation was considerably small, the participants directly matched the very specific target group of the implemented features. We take these results as an indication that this form of consent self-management does increase the perceived control of learners over their data. Therefore, also based on the agreement to the statements, that the added transparency through verification of stored personal LA data increases the trust in the processing of personal data and in the feedback generated based on the stored data, we conclude that this approach of verification can benefit the trust of learners within LA projects.

### 3.7 Discussion and Conclusion

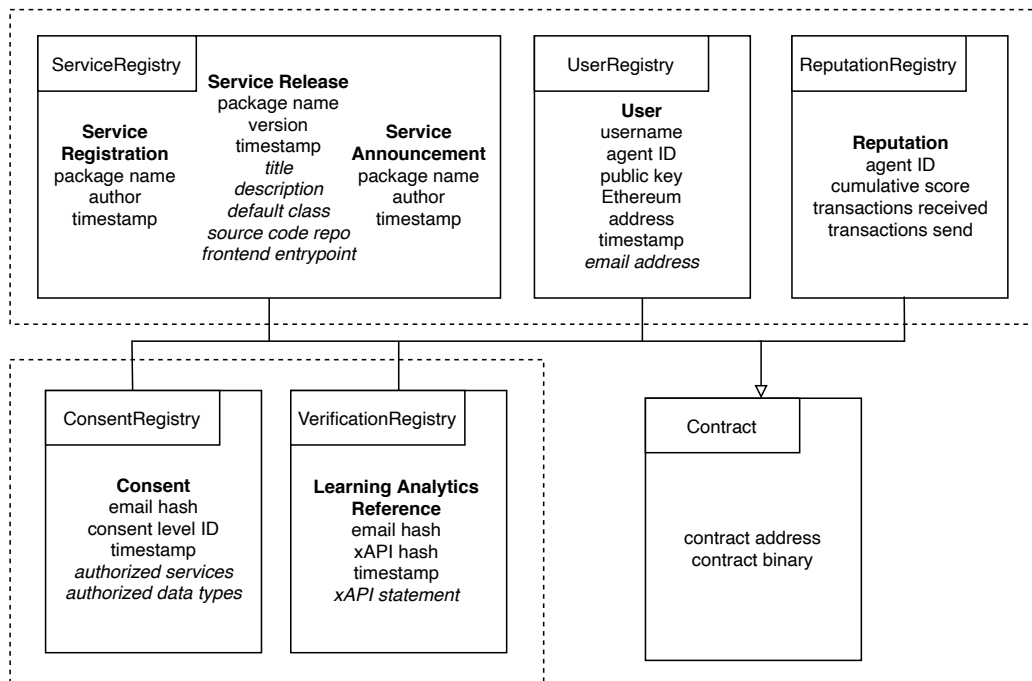


Figure 3.28: All smart contracts developed in this chapter.

This chapter introduced a decentralized infrastructure for CoPs. Based on las2peer as the technical foundation, we followed a design science approach to determine the advantages this approach bears and rigorously address the shortcomings it had. The introduced API metadata-based service discovery build the



basis for developing a blockchain-based decentralized service registry. Using smart contracts for storing both service releases and announcing currently running services for each node, this approach added verifiable browsing of services to P2P-based, decentralized CIS. To address the challenges introduced by blockchain transaction costs, we developed a smart-contract based reputation registry, that build the basis for our incentivization system. The final contribution of this chapter is the introduction of verifiable LA data collection and consent management. This blockchain-based mechanism enables learners to both manage their consent to data extraction from multimodal LA data sources, as well as the verification of their stored data. Fig. 3.28 provides an overview on all smart contracts introduced in this chapter.

With these contributions, we provided answers to our first research question “What properties does a decentralized, self-hosted infrastructure for CIS need to fulfill?” by consequently addressing sub-questions in the corresponding sections of this chapter. Now having presented the basis for CoPs to deploy and host their infrastructure, the next chapter deals with our second research question of how we can support the creation of decentralized CIS and the applications running on it.



## Chapter 4

# Scaffolding Decentralized Community Information Systems

### Summary

This chapter presents the Web-based scaffolding environment for our decentralized CIS platform. We present the enhancement of a MDWE approach with collaborative live coding and wireframing, which was then further extended to also support the model-driven creation of social learning bots. Integrated into it is also an extensive project management layer and a NRT service success evaluation suite.

**Contributions**  $\Rightarrow$  RQ 2. *Keywords:* Scaffolding; MDWE; Wireframing; Service Success Evaluation; Social Bots. The results presented here have been partially published in [LND\*16, LNKK16, LNKJ17, LNWK18, NLK119, LNRK19, LNNK20, NLK\*20]. This chapter contains partially information and content extracted from these publications.

This chapter deals with the scaffolding environment we built around the las2peer concept to support CoPs in developing their decentralized CIS. It describes the creation and evaluation of two main artifacts. The first is the *Community Application Editor (CAE)*, with its extensions of *Collaborative Wireframing Support*, the *NRT Evaluation Center* and the Web-based project management environment. The second is the *Social Bot Framework (SBF)*, which builds on top of the CAE principles to generate social bots. These create new interaction possibilities with applications (not only) running in the decentralized CIS, leveraging

communication channels already established with many CoPs.

Analog to the previous chapter, we start by describing our design science process in Sec. 4.1. Then, diverging from that structure, we introduce this chapter's main contributions in a more unified way. We start with a motivational example that builds a case for the usage of MDWE techniques for scaffolding decentralized CIS in Sec. 4.2. Sec. 4.3 describes the overall conceptual approach of the CAE and Sec. 4.4 describes its realization. We settled for this structure due to the highly interwoven parts of the single contributions made to the CAE during the course of this dissertation. Following our design science process, we present the iterative evaluations of the CAE in Sec. 4.5. Finally, as the second main contribution of this chapter, with Sec. 4.6, the SBF is introduced. Built on top of the lessons learned in developing the CAE, we present its concept, realization and usability evaluation. The real-world evaluations of applications of both the CAE (for scaffolding a decentralized CIS architecture and evaluating it) and the SBF (for providing additional interaction possibilities with the decentralized CIS) then build the subject of the next chapter.

## 4.1 Methodology

Fig. 4.1 depicts the methodology we followed in this chapter. It consists of nine iterations, of which the first four were also already included in [Nico18]. We nevertheless shortly cover them here to draw a complete picture of the research conducted in the scope of this design science process.

We started with the initial question, how to integrate end users more into development, to close the gap in requirement elicitation. This led to the development of the initial CAE prototype, which we used to redesign an existing Web application that showcased its usability. These results were communicated in a demo paper [LND\*16]. The first usage of the CAE clearly pointed out that a more defined development process was needed. Thus, we started to create the agile and cyclic development process that the CAE approach currently implements. We first evaluated this approach in multiple evaluation sessions with teams of mixed professions, as well as that we observed the usage of it within a longer timespan in a university software development lab course. Results were communicated in [LNKK16, LNKJ17]. These evaluations showed a lack of expressiveness of the modeling language for certain aspects of a Web application, which we tackled by developing the *Live Code Editor*. The results of the evaluation with student developers were published in [LNWK18].

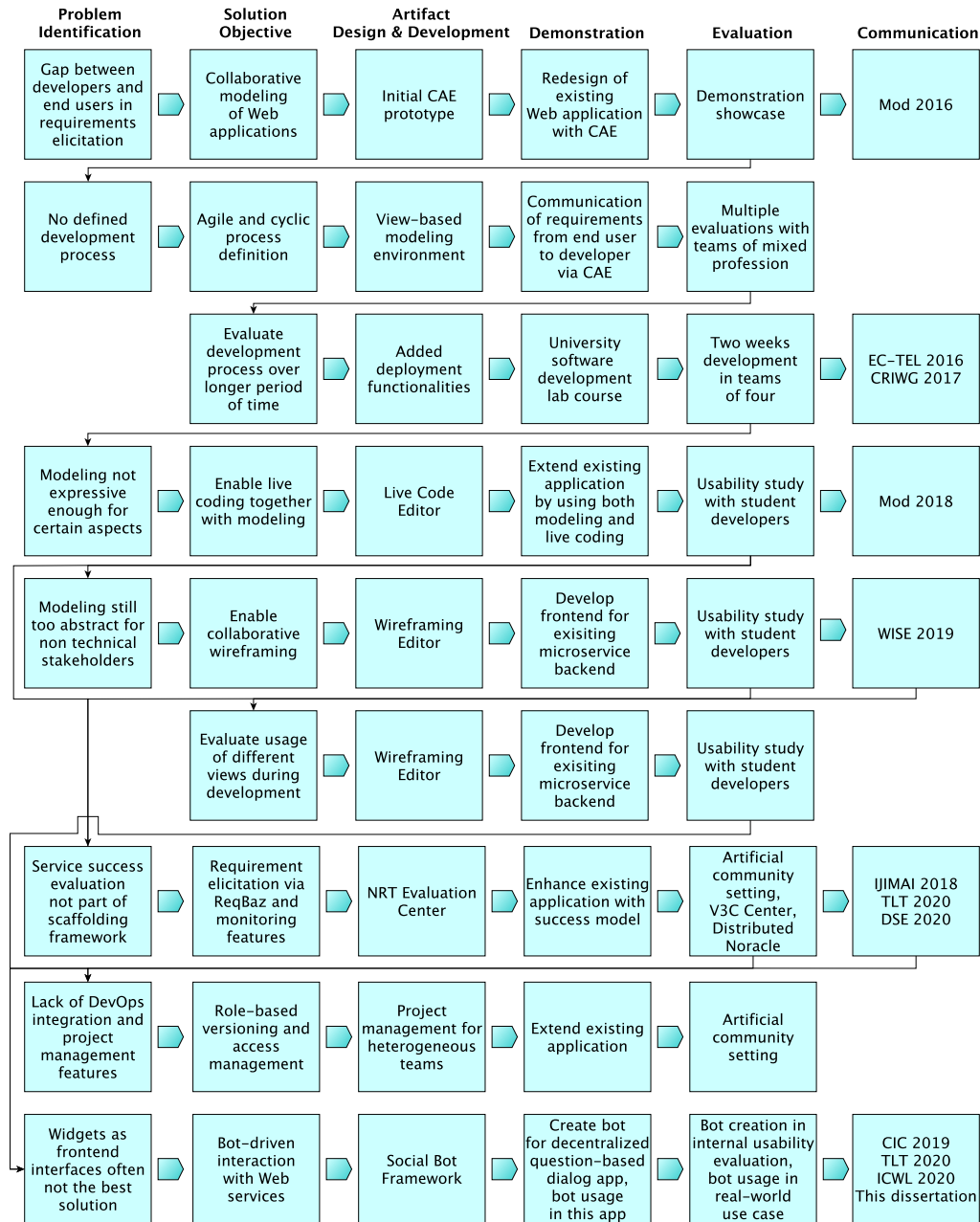


Figure 4.1: Iterations of the design science process we followed in Chapter 4.

As both the live code editor and the collaborative modeling are still rather

abstract, especially for frontend development, our next step was the integration of the collaborative *Wireframing Editor*, which we published in [LNRK19]. We continued measuring the impact of the wireframing by evaluating the time spent in different views of the CAE. To integrate the framework more closely with the overarching las2peer methodology, our next step was to integrate las2peer’s service success evaluation framework MobSOS into the CAE by making it possible to specify and integrate different service success measures already during the application’s design process. This led to the development of the *NRT Evaluation Center*, a Web-based framework that enables CoPs to collaboratively create success models for their services. It also serves as the platform for the collaborative negotiation of a service’s value for the payout of our community contribution incentivization (cf. Sec. 3.5.2). The framework was first evaluated in an artificial community setting to test its usability, before we applied it in two real-world applications and evaluations presented in the next chapter. The results were communicated in [LNNK18, LGF\*20, LNNK20]. As our environment did not feature any real project management, in the way this term is used in the DevOps world with its ever-growing tool support, we introduced a role-based project management layer to the CAE. This integrates GitHub projects, supports external dependency inclusion, semantic versioning, and commit-based history-browsing of CAE applications.

The final contribution of this chapter is the *Social Bot Framework (SBF)*, which was created with two ideas in mind. The first is that CoPs often already use conversational interfaces, and the utilization of these to connect and interact with community applications bears the potential to raise awareness for actions within the applications, even if the community member is currently not logged in. The second idea that led to the creation of the SBF concerned the development process of social bots. Here, using a model-driven approach provides the same benefits that hold true for MDWE principles: it lowers the barrier for integrating end-users into the development process, one core goal of this chapter. Results were communicated in [NLK19, NLK\*20, LGF\*20]. The combined results are published in this dissertation.

## 4.2 Motivation for Model-Driven Scaffolding

Current MDWE approaches try to increase productivity by enabling the generation of Web applications, based on information usually specified in the form of conceptual models [KMM\*08]. Corresponding to a certain domain-specific metamodel, the models reflect the structure of Web frontends and abstract the

pagination and the navigation of applications. Based on certain templates and incorporated, framework-specific best practices, the resulting applications can be specified and instantiated accordingly. By splitting the metamodel into separate views that reflect separate parts of the application, different stakeholders can focus on different parts of application design, according to their background, expertise and interest. If used in a NRT collaborative fashion, this approach bears the potential to involve non-technical stakeholders better into the development process and thereby also serves as a means to improve requirements elicitation.

However, modeling alone often cannot depict the complexity of a Web application. Certain parts of an application are very specific, and while a metamodel can enforce the overall architecture of a Web application, often manual code editing is still needed to implement the complete application functionality. To adapt to this, a collaborative MDWE approach has to support development cycles with rapid changes in the model-based architecture and the corresponding source code, both being simultaneously edited. Hence, traditional methods that enable the synchronization between model and code need to be adapted to this collaborative setting.

On the other hand, modeling (and especially manual code editing) still requires a rather good and specific development knowledge, in order to be able to model and modify the generated software artifacts. Software prototyping, often also called wireframing, is a popular software engineering method to quickly conceive the most important aspects of a software application at the early stages of software development. It is a collaborative and social process, that involves designers, end users, developers and other stakeholders. In contrast to a conceptual model, that consists of rather abstract nodes and edges, a wireframe provides a closer representation of the final Web application's visual design. Consequently, a wireframe is more intuitive and feels more familiar to non-technical stakeholders. Such an application promises a lower learning curve, with less required knowledge about Web development. In order to achieve such a novel collaborative frontend development practice, live synchronization between models and wireframes has to be implemented.

To illustrate this concept, we want to sketch a use case that integrates this novel MDWE practice. A professional community of medical doctors uses videos and images as main study and documentation objects in their training practice. We now assume that this community wants to integrate 3D objects (e.g., highly detailed digital representations of anatomical objects) in their training practices. Such features cannot be easily implemented without technical knowledge. On the contrary, they are also hard to explain to developers without deeper domain

knowledge. Thus, the community uses a Web-based MDWE approach for requirements elicitation with (possibly external) developers. Doctors and developers can now distribute according to their domain-specific knowledge to work on the corresponding views. For example, doctors could produce wireframes to explain the developers their proposed extension of the current system. Directly transforming these wireframes into models, developers start working on the corresponding models and source code, all directly in the browser and in NRT. At all times, the Web application is automatically generated and deployed on the Web, thus the community can follow along and provide direct feedback on the current state of the prototype, via an integrated success evaluation suite.

The Community Application Editor (CAE) was developed to fulfill these requirements. It integrates both *Live Code Editing* and *Collaborative Wireframing*, all in a NRT collaborative setting on the Web. Source code is directly pushed to GitHub, a *Project Management* layer provides user- and component management specifically tailored to the needs of CoPs, and the CAE integrates with the overarching development methodology of las2peer by integrating the *Requirements Bazaar* [RBKJ13] and *MobSOS* directly from within the Web-based environment, unifying them under the new *NRT Evaluation Center* app. In the following, we present the concept, realization and evaluation of these works.

## 4.3 Conceptual Overview

In this section, we provide an overview on the conceptual underpinnings that we followed when realizing the CAE. We start with a description of the different views and phases that make up our scaffolding environment in Sec. 4.3.1. Next, we introduce CAE's Web application metamodel in Sec. 4.3.2, before we come to the role-based project management layer of the CAE (Sec. 4.3.3). We continue with the conceptual integration of the *NRT Evaluation Center* (Sec. 4.3.4) within our scaffolding environment. Then, shortly the live code editor integration is covered in Sec. 4.3.5, before we describe in detail the integration of the wireframing support (Sec. 4.3.6).

### 4.3.1 View-based Model-Driven Web Engineering

Our approach follows the separation of concerns principle [Kent02] and defines four orthogonal views [NRD\*18] for the modeling of Web applications, based on a comprehensive metamodel:



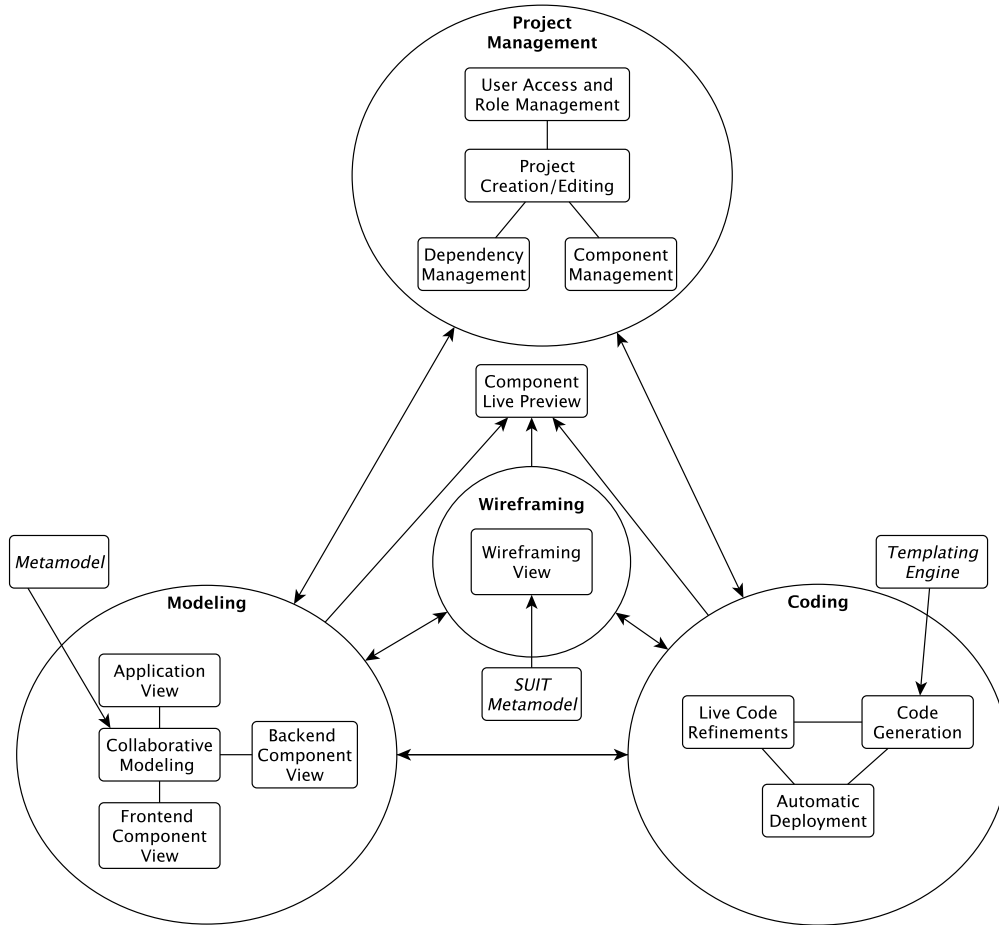


Figure 4.2: Overview of the MDWE approach.

- The *Frontend Component View*, represented by a model of a Web component.
- The *Backend Component View*, represented by a model of a microservice.
- The *Wireframing View*, a visual representation of a frontend component.
- The *Application View*, the metamodel of the complete Web application.

The development process itself can be split up into four main phases, namely the *Modeling*, the *Coding*, the *Wireframing* and the *Project Management* phase. Fig. 4.2 gives an overview of these development phases, their corresponding

views, as well as their connections with each other. Based on this modeling – coding – wireframing – project management cycle, the approach enables the cyclic, collaborative, model-driven creation of Web applications.

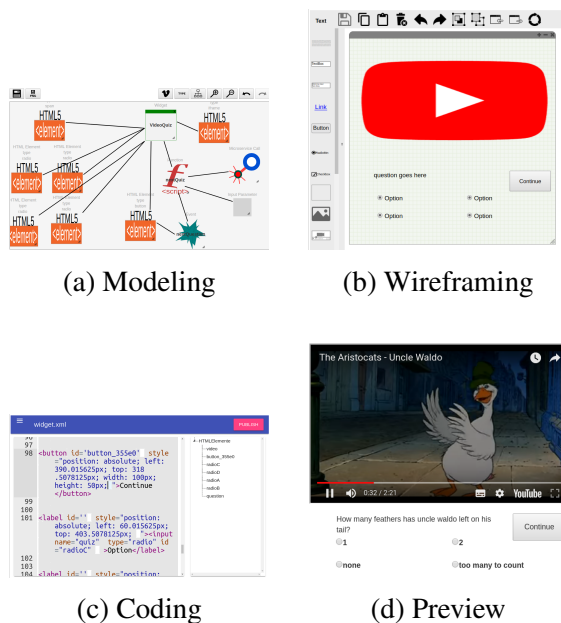


Figure 4.3: Different representations of the same frontend component.

To illustrate this concept, Fig. 4.3 depicts four representations of the same frontend component. Fig. 4.3a shows the conceptual model. Fig. 4.3b depicts the wireframe visualization of the frontend component model. Both the wireframe model and the conceptual model are used as input for the code generation to generate the code artifacts depicted in the Live Code Editor of Fig. 4.3c. Finally, Fig. 4.3d shows a live preview of the resulting application, based on the generated code artifacts.

### 4.3.2 Web Application Metamodel

Although our general approach could be used for arbitrary MDWE frameworks and Web applications, in the scope of this dissertation, we consider Web applications composed of HTML5 and JavaScript frontends, and RESTful microservice backends. Fig. 4.4 depicts the Web application metamodel. The central entity of a microservice is a *RESTful Resource*. It contains *HTTP Methods*, which form

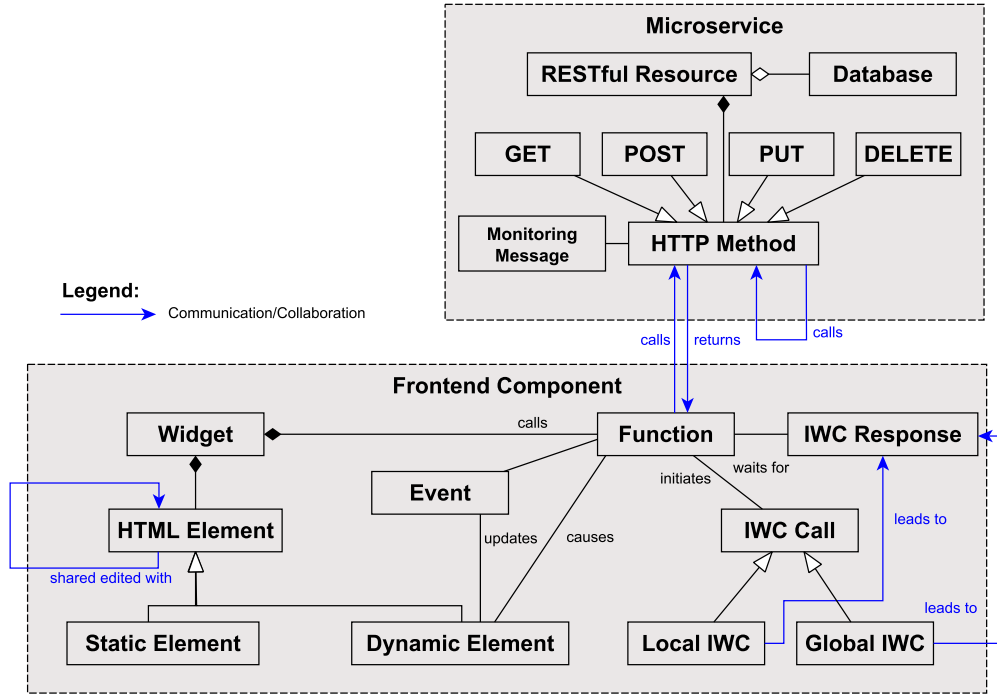


Figure 4.4: The underlying Web application metamodel used in the CAE.

the interface for communication either via a RESTful approach, but also via an *Internal Service Call* from one HTTP method to another, possibly between different microservices. To enable service monitoring, each HTTP method can be enhanced with multiple *Monitoring Messages*. According to the idea of polyglot persistence, each microservice can have access to its own *Database* instance.

The central entity of a frontend component is a *Widget*. This widget consists of *Functions* and *HTML Elements*. HTML elements can either be *static*, meaning that they are not modified by any other element or functionality of the component, or *dynamic*, meaning that they either are created or updated by one of the frontend component's elements. Both static and dynamic HTML elements can trigger events, which can for example be a mouse click, that cause function calls. The second option to trigger a function call is via an *Inter Widget Communication (IWC) Response* object, that waits for an *IWC Call* to be triggered. These calls are again part of a function, which initiates them. A function is able to update or create a dynamic HTML element. The last part of the frontend component view is the communication and collaboration functionality, which includes the already

mentioned IWC call - response mechanism, as well as microservice calls that are triggered by a function. HTML elements can also be instrumentalized with collaborative support, making it possible for elements to share the same state in the Web browser of all participating users, propagating changes in NRT.

### 4.3.3 Role-based Project Management

To enable multiple CoPs to collaborate on multiple community applications at the same time, the CAE contains an extensive project management layer<sup>1</sup> on top of the four modeling, coding and wireframing views. This layer has two main features. The first is access management, which enables a CoP to restrict write access to certain projects, as well as to assign certain roles to community members. The second is component and dependency handling, which allows for versioned management of frontend and backend components of a community application. Additionally, the project management layer also integrates the CAE deeper with current DevOps development practices, by supporting GitHub's issue management environment directly from within the UI of the CAE. This allows using a Kanban board to plan which development tasks should be done and to inform every project member on the current development state.

Fig. 4.5 shows the Entity Relationship (ER) diagram of the CAE. As one can see, the connection to external services, namely the *Requirements Bazaar* and *GitHub* are depicted in green, while those entities that are metamodel-based are depicted in blue. The latter are stored in the database following the format described in Sec. 4.3.2, or the corresponding wireframing SUIT model, which will be introduced in Sec. 4.3.6. Since the source code corresponding to a model representation is solely stored on GitHub, it is not reflected in this diagram.

The central entity of the CAE is a *Project*. It belongs to a CoP and thus is created and maintained by its members, which can have different *Roles* within the development process. Note, that it is possible for a *User* to have different roles in different projects and/or communities. Users have their *GitHub Account* connected to the CAE to be able to access GitHub's issue management environment. Each project in the CAE is connected to a *GitHub Project*. Initially it is initialized with a simple Kanban board containing the three columns "To do", "In progress" and "Done".

Projects consist of several *Applications*. While from a practical point of view,

---

<sup>1</sup>The prototype implementation was partially supported by a bachelor thesis [Doli20] under supervision of the author.

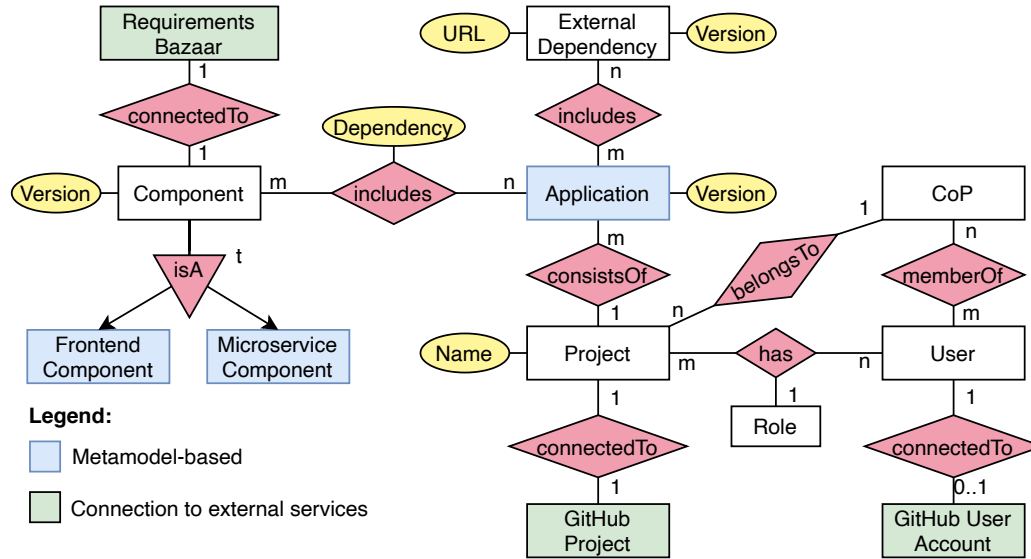


Figure 4.5: ER diagram of the CAE.

there exists only one application for each project, each application's version is treated as a new application within the scope of this ER diagram. Each application includes several *Components*. These versioned components each include a *Requirements Bazaar* connection, depicting a category in that project representing the component. Components are either *Frontend*- or *Microservice Components*, corresponding to the CAE metamodel. If a component of another project is used, it is included as a dependency, making it immutable from within the modeling environment. Besides using components of other projects as dependencies, there is also the possibility to include *External Dependencies* from a GitHub repository. We restrict these here to Web widgets for frontend- and las2peer services for microservice dependencies, such that they can be easily embedded in the deployment process (see Sec. 4.4).

After a project was created, members of the CoP can invite each other via an invitation mechanism, using their OpenID Connect (OIDC) username. As a list of all project members is shown, the complete CoP is aware of their members which are currently working on the project and its components. This increases the social awareness (or rather the availability awareness [XhPo10]), since a user receives information on the presence of other people in the shared environment. Following the inclusive spirit of OSS, users who are not members of the CoP can still view the project and its components (although without editing rights) before joining the

community.

As already stated, the CAE features a role-based access management. Based on a user's role, view elements of the CAE are highlighted or hidden. This allows the four views of the CAE to be tailored more specifically to the individual user. With this, we aim at reducing the deterrence of non-technical users, because the UI elements that require more technical expertise can be hidden from them. The UI element highlighting and hiding mechanism improves the guidance of users through the MDWE process of the CAE, as it helps them to focus on the UI elements that are of importance to them. Each user gets assigned one role per project and every role is linked to a configuration of UI elements which is shown to users with this role assigned. The assignment of roles to users and the configuration of the UI elements is done in the project management layer. These roles are fully customizable by all community members.

#### 4.3.4 A NRT Evaluation Center

What metrics are important and how to calculate a service success score varies from community to community and from service to service. These parameters can also change over time within the same context. Thus, evaluation parameters must be constantly adapted not only to a changing implementation, but also to a constantly developing understanding of the term success within a CoP. MobSOS, as the success evaluation framework of las2peer, takes these changing requirements into account by providing custom tailored service success models for each community service and each community. Following the “observe, where possible; only survey, where inevitable” idea [RKJa15], MobSOS also contains a module for creating questionnaires, called *MobSOS Surveys*, which the community can use to evaluate those parts of their service that cannot easily be measured from within the service itself.

However, although MobSOS allows non-technical users to see visualizations of success dimensions like quality and impact, it relies on service developers to directly code the measures into their services, and upload service success models in a rather technical format to a las2peer node directly. There exists no mechanism to discuss the success factors within the community. Furthermore, MobSOS Surveys is not directly integrated into the service success model. There is no graphical UI for reviewing or editing the processes behind the community success evaluation. Thus, the community must rely on developers to correctly comprehend and implement the community's understanding of success. If developers fail to do so, the success evaluation results are distorted and a successful service might be

discontinued or an unsuccessful community service might be kept alive.

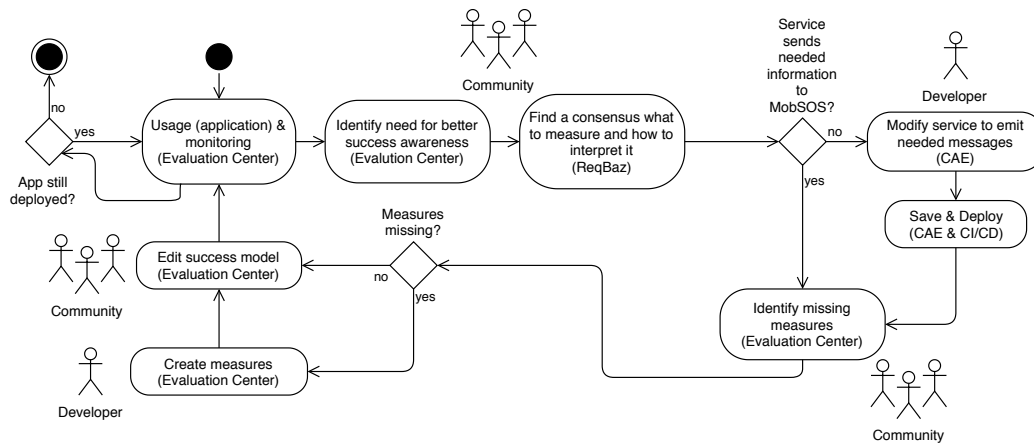
This lead us to the development of the *NRT Evaluation Center*, a Progressive Web App (PWA) that bundles the services MobSOS provides<sup>2</sup>. It offers a Web-based user interface enabling the end user to create and maintain success evaluation, all in NRT. The overarching goal was enabling communities to define and continuously adapt the success evaluation process in las2peer with as much involvement of non-technical community members as possible. We approached this goal by integrating the success evaluation closer into the development process of the CAE. The direct link to the Requirements Bazaar, which we introduced with the project management in the previous section, enables the discussion of not only the service (model), but also its success model in an end user approachable way. We also extended MobSOS to integrate a survey's questions within the service success model itself.

Additionally, we extended the CAE's microservice metamodel with modeling elements for different types of *Monitoring Messages*. Specifically, *Monitoring Messages* can be connected to an *HTTP Method*, which then logs its processing time, to an *HTTP Payload*, which logs the payload content, or to an *HTTP Response*, to log the response content. These measures are available to be extended in the Live Code Editor or to be directly used in the service success modeling. These additions led to an integrated service success evaluation flow that is tightly integrated within the CAE's development methodology, implementing core ideas of the DevOpsUse [LNKK16, Kore20] life-cycle.

Fig. 4.6 depicts this integration of the NRT Evaluation Center into the development process of the CAE. The process starts with the decision, if the app is still used. If so, the application is monitored during its usage, using the NRT Evaluation Center, until the need for better or changing success awareness is identified. This decision, as well as the consensus of what do measure and how to interpret these measures is discussed in the Requirements Bazaar, either directly from within the NRT Evaluation Center, or from the Requirements Bazaar interface. Once consensus is reached, it has to be evaluated, if the service in question already sends the needed monitoring data to MobSOS. If that is not the case, the CAE can be used to modify the service's model and source code to emit the needed data. The process continues with the CoP identifying and modifying/creating the missing measures collaboratively in the NRT Evaluation Center. Finally, the circle starts again with the usage and monitoring of the application with the now-updated

---

<sup>2</sup>The prototype implementation was partially supported by a master thesis [Hoss19] under supervision of the author.



success evaluation model.

### 4.3.5 Model Synchronization for Live Code Editing

We unify the architecture of applications developed with our approach through the usage of protected segments that enforce a certain base architecture, facilitating future service and frontend orchestration, maintenance and training efforts for new developers<sup>3</sup>. Protected segments in the source code describe a functionality that is reflected by a modeling element. In order to encourage the reuse of software components, we allow changes which modify the architecture only in modeling phases. Since our approach offers a cyclic development process, this can be done instantly by switching to modeling, changing the corresponding element and returning to a new coding phase. To further enforce this methodology, before source code changes are persisted, a model violation detection is performed. This informs the user about source code violating its corresponding model, e.g., architecture elements manually added to the source code instead of being modeled. Concerning the synchronization between the code and the model, our collaborative MDWE process uses a trace-based approach. Changes in the code produce traces, which are used in the model-to-code (re)generation in order to keep the corresponding code synchronized to the model elements. This way, the process can be reflected without the need to implement a full Round-Trip Engineering (RTE) approach.

<sup>3</sup>The prototype implementation was partially supported by a bachelor thesis [Wink16] under supervision of the author.



More information on the realization of the model to code synchronization strategies of the CAE can be found in [Nico18, LNWK18, LNNK20].

### 4.3.6 Integration of Wireframing Support for MDWE

Inspired by the concepts of MockupDD and following the view separation of ArchiMate (both presented in Sec. 2.5.2), we developed a SUIT model for the wireframing integration<sup>4</sup> and defined the transformations of this model to the frontend component metamodel, depicted in Fig. 4.7. The SUIT model of the

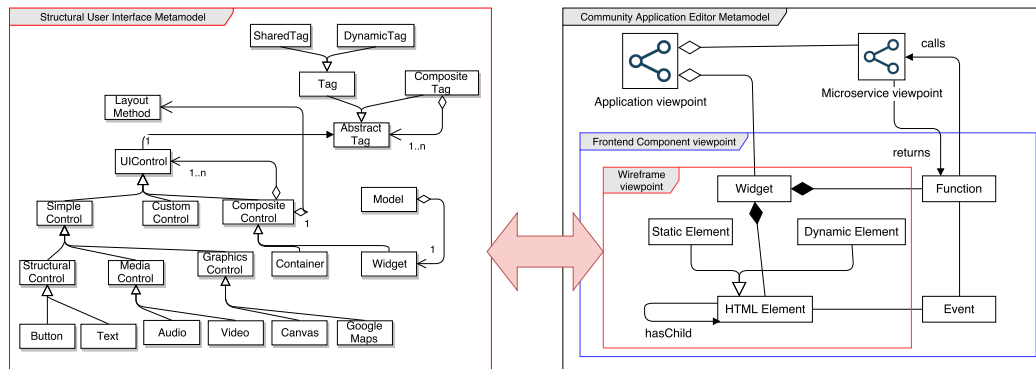


Figure 4.7: Mapping of the SUIT- to the MDWE metamodel.

wireframing editor comprises the most common HyperText Markup Language (HTML) elements of the current HTML5 standard. It offers simple structural elements like buttons, text boxes and containers. Furthermore, media elements like the HTML5 video and audio player and custom Polymer elements<sup>5</sup> are supported. Also compositions of elements are defined, like a checkbox with a label. Each UI control element of the SUIT model has its own set of attributes defined, according to the HTML5 standard. We also introduced a so called 'SharedTag', that can be assigned to any UI control element to add NRT collaborative behavior to it.

Conceptually, an instance of the SUIT model is a labeled tree. We formally define such a tree as a connected, acyclic and labeled graph. An arbitrary element  $v \in V$  always has the signature  $v = l, t, A$ , where  $l \in \Sigma$  is the label, with  $\Sigma$  being a finite alphabet of vertex and edge labels.  $t \in T$  is the type of the node, where  $T$  is either a UI control element or a tag defined in the SUIT metamodel, as

<sup>4</sup>The prototype implementation was partially supported by a master thesis [Rose17] under supervision of the author.

<sup>5</sup><https://www.polymer-project.org>

depicted in Fig. 4.7. For example  $T$  might consist of the following elements:  $UI = \{Text, Button, Video, Canvas, ..\}$  and  $Tag = \{SharedTag, DynamicTag, ..\}$  with  $T = UI \cup Tag$ .  $A$  is a finite set of properties related to an UI control element or tag and each  $a \in A$  is a key-value-pair  $k, v$  with  $k, v \in \Sigma$ . The tree always consists of a distinguished vertex  $r$ , which is also called the root. The root is always of type *Widget*. The *parentv* function is a helper function that yields the parent-vertex for a vertex of the SUIT tree. If the vertex  $v$  is the root, the root will be returned.

**Definition 1** A SUIT model is a labeled tree with  $SUIT = V, E$ .  $V$  is a finite, non-empty set of vertices.  $V$  is always initialized with the root  $r$ .  $E$  is a set of unordered pairs of distinct vertices  $v1, v2$  with  $v1 \neq v2$ , which constitutes the edges of the tree.

For the integration into our MDWE approach, a SUIT model is mapped to an instance of the frontend component view. Let  $VP = V, E$  be an acyclic, directed graph that represents an arbitrary view. An edge  $e \in E$  of such a graph has the signature  $l, t, v1, v2, A$ , where  $l \in \Sigma$ ,  $t$  is the type of the edge,  $v1, v2 \in V$  and  $A$  is a set of key-value-pairs that constitute the attributes of the edge.

**Definition 2** An instance  $M$  of a view of  $VP$  is an acyclic, directed graph with  $M = V', E'$ . For each  $v \in V'$  holds  $typev \in labelV$ , with *type* and *label* being helper functions defined as:

$type : V \mapsto \Sigma : l, t, A \mapsto t$  and  $label : V \mapsto \Sigma : l, t, A \mapsto l$ .

Analogously, these functions are defined for an edge  $e \in E'$  of a view.

Now let  $VP_{wireframe}$  be the acyclic directed graph representing an arbitrary instance of the wireframe view and  $W_{SUIT}$  a SUIT model representing a concrete wireframe. An instance of the SUIT model is mapped to an instance of the wireframe view with function  $\varphi$ :

$$\varphi : W_{SUIT} \mapsto VP_{wireframe} = V, E \mapsto \phi V, \gamma E$$

where  $\phi$  is defined as follows:

$$\phi V = \{\phi v | v \in V\} \text{ with}$$

$$\phi : V \mapsto V' = l, t, A \mapsto l', t', A' :$$

$$\begin{cases} l, t, A \mapsto l, \text{HTML Element}, & \text{for } t \in UI \\ A \cup \{\text{type}, t, \text{static}, \text{true}, \text{collaborative}, \text{false}\}, & \\ l, \text{SharedTag}, \emptyset \mapsto l', \text{HTML Element}, \text{shared}A', & \text{for } l' = \text{parent}l \\ l, \text{DynamicTag}, \emptyset \mapsto l', \text{HTML Element}, \text{dynamic}A', & \text{for } l' = \text{parent}l \\ l, \text{Widget}, A \mapsto l, \text{Widget}, A, & \text{otherwise} \end{cases}$$

where *shared* and *dynamic* are functions that are applied to every attribute in *A* of the referenced 'HTML Element'-node. These helper functions change the value of the 'collaborative'- respectively 'static'-attribute for the referenced 'HTML Element'-node. All other attributes are left untouched. Thus, *shared* is defined as

$$\text{shared}A = \{\text{shared}'a | a \in A\}$$

with

$$\text{shared}' : A \mapsto A : \begin{cases} k, \text{false} \mapsto k, \text{true}, & \text{for } k = \text{collaborative} \\ k, v \mapsto k, v, & \text{otherwise} \end{cases}$$

and *dynamic* is defined as

$$\text{dynamic}A = \{\text{dynamic}'a | a \in A\}$$

with

$$\text{dynamic}' : A \mapsto A : \begin{cases} k, \text{true} \mapsto k, \text{false}, & \text{for } k = \text{static} \\ k, v \mapsto k, v, & \text{otherwise.} \end{cases}$$

The relationships between the nodes in the wireframe view are generated with  $\gamma$ :

$$\gamma E = \{\gamma e | e \in E\}$$

with

$$\gamma : E \mapsto E' = v_1, v_2 \mapsto l, t, v'_1, v'_2, A : \begin{cases} v_1, v_2 \mapsto l, \text{Wid. To El.}, v_1, v_2, A, & \text{for } v_1 = r, \text{type}v_2 \in UI \\ v_1, v_2 \mapsto \{l, \text{hasChild}, v_1, v_2, A\}, & \text{for } \text{type}v_1 \in UI, \text{type}v_2 \in UI \\ & \text{and } v_1 \neq v_2 \neq r \end{cases}$$

With  $\varphi$ , we only map the UI elements of the SUIT model to the wireframe view. An 'HTML element' node of the frontend component-, respectively wireframe view, consists of the four properties id, type, static and collaborative. The id of the HTML element is automatically generated by the mapping approach. The value of the type attribute is an element from the UI. The static and collaborative attributes are the only attributes represented as tags in the SUIT model. Furthermore,

they are simple Boolean attributes and therefore have no own attributes defined. Additionally, the tags are unique and thus they only appear once for a certain UI element.

A node of the SUIT tree is mapped with  $\phi$  to a certain 'HTML Element' or 'Widget' node. An arbitrary UI element of the SUIT model is always mapped to an instance of the 'HTML Element' node class, where the label of the UI element is the label of the node. The type of the UI element is mapped to the type-attribute of the node. By default, the 'static' attribute is true and the 'collaborative' attribute is false. To change the values of these attributes, a DynamicTag- respectively SharedTag element is mapped to the corresponding attribute in the 'HTML Element' node. For each tag a function is required, which alters a certain aspect of the signature of an 'HTML Element' node (e.g. type or attribute). For the definition of the current mapping approach, the two helper functions *shared* and *dynamic* are defined, which change the Boolean value of the associated attribute. The root-element of the SUIT tree is always mapped to the 'Widget'-node, where the label of the root is also the label of the 'Widget'-node. The same holds for the attributes. With function  $\gamma$ , the relationships between nodes are generated. The function comprises two cases. If the UI element is a direct child from the root, a single 'Widget To HTML Element' edge is created (abbreviated in the function with 'Wid. To El.', due to space restrictions). For the second case, we assume that  $v_1$  is a parent of  $v_2$  and  $v_1$  and  $v_2$  are not the root. Then, the 'hasChild' relationship is generated.

## 4.4 Realization

In this section, we present the user interface of the overall CAE (Sec. 4.4.1), with a special focus on the model versioning system (Sec. 4.4.2). We continue introducing CAE's general architecture (Sec. 4.4.3) and conclude with the integration of the wireframing (Sec. 4.4.4).

### 4.4.1 User Interface

Fig. 4.8 shows a screenshot of the frontend component modeling space of the CAE. The *Wireframing View* is depicted in the upper-, the *Modeling Canvas* in the lower-left, while the *Live Code Editor* can be found in the lower-right. The upper-center depicts the *Live Preview* and the selected modeling element's *Property Window*,

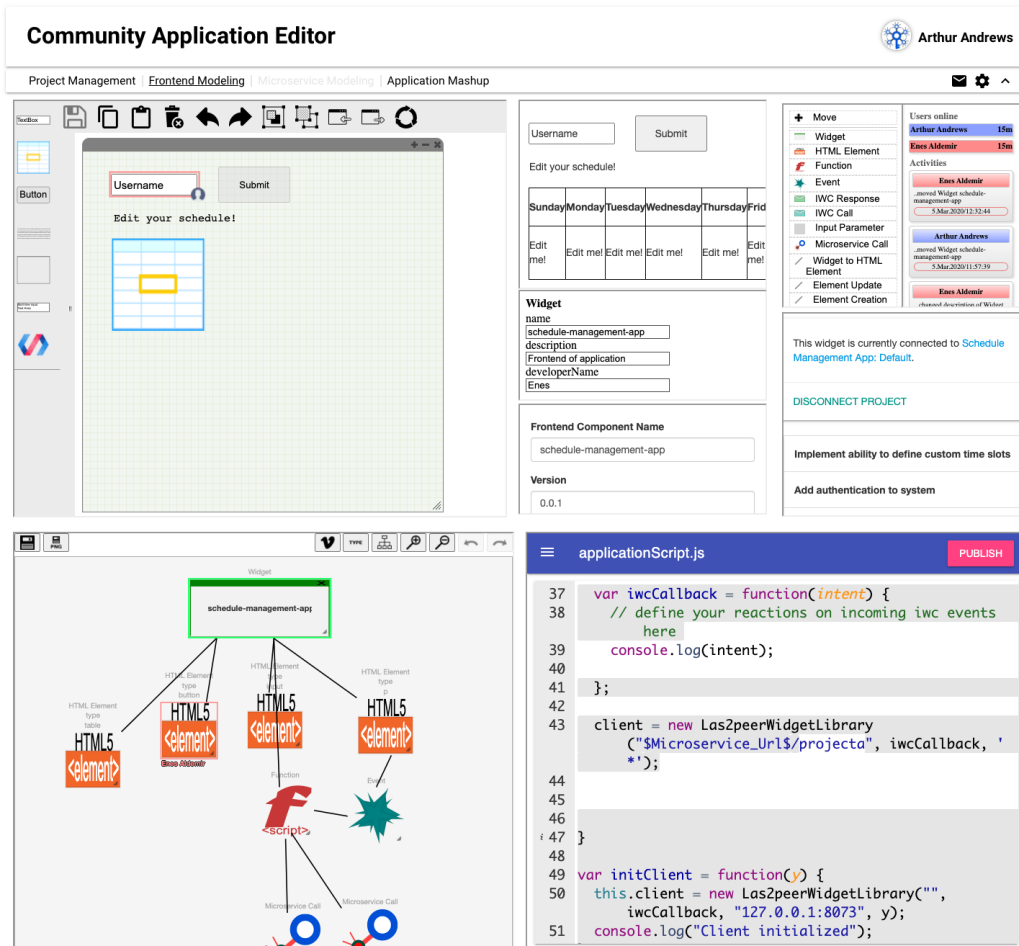


Figure 4.8: Screenshot of the frontend component modeling space.

and on the outer right the *Modeling Palette*, *Activity Widget* and *Requirements Bazaar Integration* are displayed.

The CAE supports common utility functions, like copy&paste, deletion of an arbitrary number of selected elements and an undo&redo functionality for both the Modeling Canvas, Live Code- and Wireframing Editor. It uses an automatic save functionality: each altering in the editor saves the current state of all models to the shared editing framework in the Yjs shared data space (cf. Sec. 4.4.3).

The editor provides awareness features to support the collaboration. The activity widget shows all collaborators currently working in one of the different views. If a remote user selects one or more elements, each element is highlighted

with a surrounding frame and marked with the image of their OIDC profile they used to log into the CAE. This holds for both the modeling canvas, as well as for the wireframing view and the live code editor.

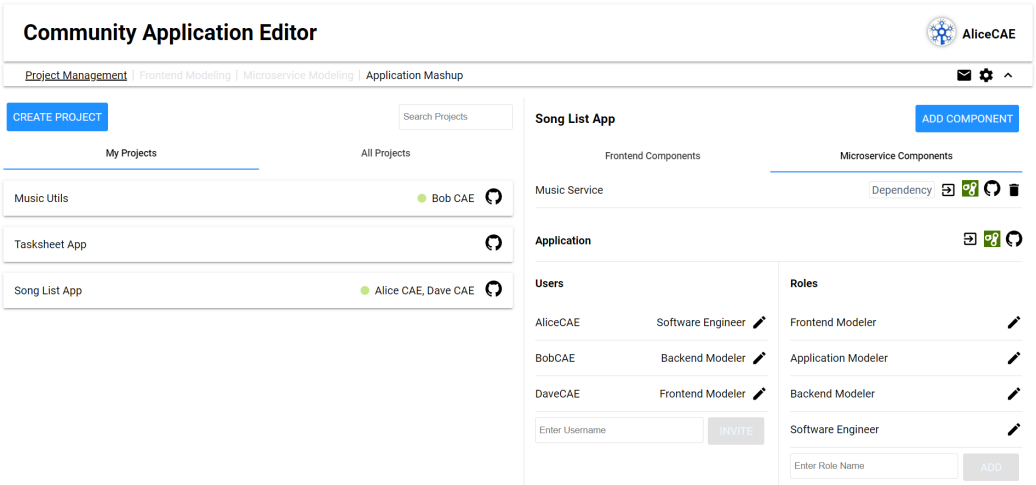


Figure 4.9: Screenshot of the CAE project management features.

Fig. 4.9 shows a screenshot of the project management layer of the CAE. The left side gives an overview about those projects the user is assigned to and also provides the possibility to browse all existing projects. For those projects the user is a member of, the interface also shows other members of the community currently online and working on these projects. The right side provides details of a currently selected project. This includes a listing of all users and their roles, as well as the possibility to change those. A link to the corresponding Requirements Bazaar category and to the corresponding GitHub project can be found here as well. Here, it is also possible to create new components or to enter the development space of already existing components of a project.

Fig. 4.10 shows two (partial) screenshots of the NRT Evaluation Center PWA. The left side shows the main overview of a success model for an application. On the top, one can enable the editing mode and invite new users to the success modeling group, and also accept incoming invitations. Below, both the option to connect a questionnaire from the MobSOS Surveys module, as well as a success dimension (“System Quality”), with its connected measure (“Performance”) is shown. The right side shows parts of an exemplary success model for a “Mensa App”. Here, the visualizations and Key Performance Indicators (KPIs) that were collaboratively defined in the NRT Evaluation Center are visualized.

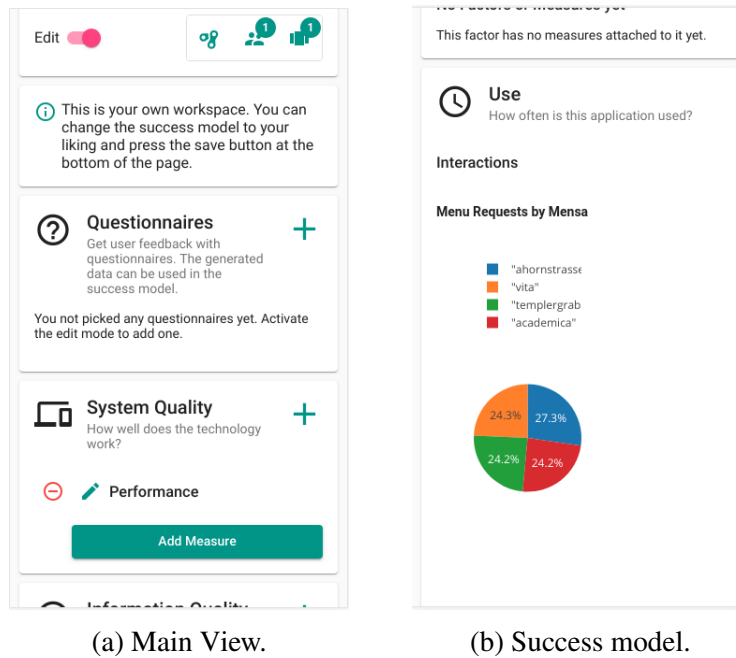


Figure 4.10: Screenshot of the NRT Evaluation Center PWA.

#### 4.4.2 Versioning System

To provide an overview on the changes in the modeling, wireframing and live coding, and to allow reverting these changes, as well as to version the individual stages of a component, the CAE contains a versioning system. As in Git, this versioning system is commit-based. Therefore, both the commit messages and version tags entered in the CAE are reflected in a corresponding Git repository.

Fig. 4.11 shows two screenshots of the versioning widget, depicting the changes of a microservice component. The left screenshot shows a list of previous commits made to the component and the details of the selected commit. As one can see, it depicts both the commits automatically pushed to the Git repository by the live code editor, as well as manual commits, created by a community member. For each commit, the modeling elements are listed, together with the changes done to the attributes of each element. The versioning system allows to load the model and wireframe of a previous commit, by just clicking on an item in the commit list. The canvas widget also moves to the position of the highlighted element so that the user does not have to search for it in the model. Besides that, the versioning widget allows to roll back the changes which were done since the last commit.

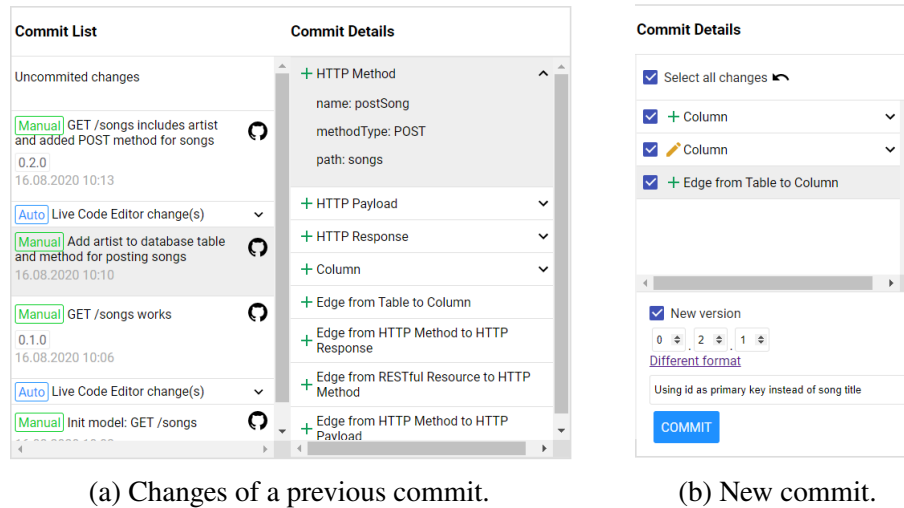


Figure 4.11: Screenshots of the versioning widget.

The right screenshot shows the creation of a new (manual) commit. Here, the user can select, which changes that should be included in this commit. All changes that are not reflected in the commit will of course still be stored in the current version of the component (within the shared frontend data space), but will not be reflected in the corresponding Git repository and relational database that manages the different versions of a model. A commit saves the changes to the component and can be named. Besides that, it is possible to tag a commit with a version number, which then also gets set as a tag on the corresponding Git repository. The users are nudged to use the *Semantic Versioning* format [RDVi14]. When a user updates a field of the semantic version number, a popup appears that asks the user to verify that the version change is justified. This should ensure that the semantic versioning format is used correctly. Following a nudging approach [KoHe15], different versioning formats are still allowed.

### 4.4.3 Architectural Overview

Fig. 4.12 provides an overview of the complete architecture. Our frontend is composed of HTML5 Web components and (apart from the wireframe model, which uses an Extensible Markup Language (XML) representation) uses JSON representations of the models. We use a lightweight meta-modeling framework, called SyncMeta [DNE\*15], to realize the collaborative modeling functionalities



of the CAE. It supports NRT collaborative modeling by using Yjs [NJDK16], a Conflict-free Replicated Data Type (CRDT) framework. For communication with the backend, we use a REST API. The backend, realized as a las2peer

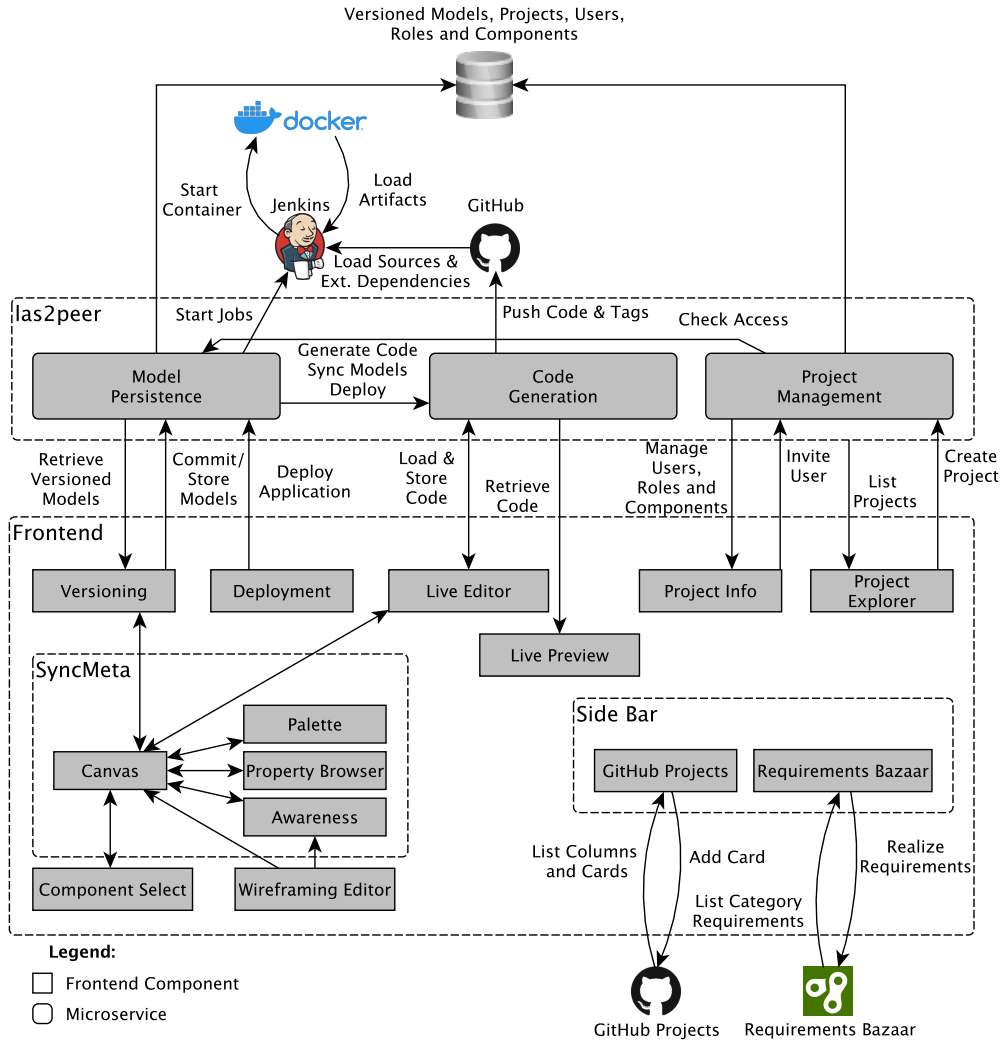


Figure 4.12: Architecture of the CAE.

network, is composed of three services. The *Model Persistence* service manages the persistence of the microservice- and frontend component models (together with their enriching wireframe SUIT models, if existing, see Sec. 4.4.4) in a relational

database. A template engine forms the main component for trace generation and model synchronization. It is used for both the initial code generation, as well as for further model synchronization processes [Nico18, LNNK20]. The *Code Generation* service implements both the synchronization with the trace models, as well as it is responsible to generate the resulting source code from the models and trace models. The source code is directly pushed to a GitHub repository, using commit messages to create a history of the modeling process for later reference. We use a Jenkins - Docker continuous deployment pipeline to deploy the resulting services in a las2peer network, directly from the modeling environment. Finally, the *Project Management* service manages on a backend level the write access to different components and provides project metadata to the corresponding frontend components via the REST API.

#### 4.4.4 Wireframe- and Frontend Component Model Transformations

Wireframe- and frontend component models are persisted next to each other, as the SUIT model enriches the HTML elements of the frontend component model with additional metadata and type-specific attributes. Thus, for code generation, a frontend component model is always required, while the SUIT model is optional. A *Wireframe to Model Transformation* and a *Model to Wireframe Transformation* were developed to transform the SUIT wireframe model to a frontend component model and vice versa. The two transformations are only needed, if one of the two frontend component representations is not existing. After that, the two model states are kept synchronized by the *Live Mapper*.

**Wireframe to Model Transformation** The wireframe to model transformation takes as input an instance of a wireframe model and the frontend component metamodel. The output of the transformation is a JSON object of the frontend component model. The implementation uses templates of a node-, edge- and attribute-representation in JSON of the frontend component model. First, the transformation algorithm generates the 'Widget'-node, which represents the root element of the frontend component model. Then, it recursively traverses the wireframe model and creates a corresponding 'HTML Element'-node for each UI control element. The 'type'-attribute of the node is set to the value of the 'HTML Element'-node name of the corresponding UI control element. Furthermore, the 'HTML Element'-node is marked as static and the 'id'-attribute of the node is

automatically generated. The value of the id is composed of the 'type'-attribute value and unique. The identifier of the UI control element is reused for the resulting node, which allows to trace back an 'HTML Element'-node to the UI control element. This is necessary for the awareness features and the live mapper. For each node, a 'Widget to HTML Element'-edge is generated, because each node has a connection to the 'Widget'-root node. If the parent of the UI control element is not the root, additionally a 'hasChild'-edge is added to the set of edges. This edge type denotes the hierarchical structure of the wireframe. It connects the parent UI control element to one of its child elements. If a UI control element has the 'shared'-tag assigned to it, the 'collaborative'-attribute of the corresponding 'HTML Element'-node is set to true as well. Since the frontend component metamodel allows every HTML Element to be collaborative, the wireframing editor allows this as well. The result of this transformation is a valid instance of the frontend component metamodel. However, the HTML attributes specified for a certain UI control element are lost, because the frontend component metamodel does not offer a way to represent them. Additionally the width, height and position of the UI control element in the wireframing editor are not related in any way to the position and dimension of corresponding 'HTML Element'-node. Therefore it is necessary to apply an auto-layout for directed graphs to the model, so that it is displayed correctly in the modeling canvas.

**Model to Wireframe Transformation** The input for this transformation is a JSON representation of the frontend component model and an instance of the wireframe editor. The latter is required to map the 'type'-attribute of an 'HTML Element' node to the correct UI control element. Since the wireframe only represents the HTML elements of the frontend component model, we only have to consider the 'Widget' node (for the size of the whole frontend component) and those 'HTML Element' nodes that are connected to the 'Widget'-node and marked as static. All other node and edge types of the frontend component model can be ignored for this transformation. As already described in the previous transformation algorithm, certain UI layout information (for example the size and position of elements) is not present in the frontend component model. Thus, we initialize these attributes with default values defined in the wireframe model. Finally, the transformation algorithm assigns the 'shared'-tag to every 'HTML Element'-node which has the 'collaborative'-attribute set to true. The result of the transformation approach is an XML document that represents the wireframe model. The resulting model is then stored in the shared data space alongside with the frontend

component model.

**Live Mapper** The live mapper listens to events of the Modeling Canvas of the frontend component modeling view and to the Wireframing Editor. In contrast to the two previously described transformations, the live mapper directly applies changes to the wireframe and frontend component model and visualizes the results in NRT. Additionally, the live mapper provides awareness features for the selection of entities on both the Modeling Canvas and the Wireframing Editor. To give an example of the live mapping, the creation of a button element in the Wireframing Editor leads to five to six operations on the Modeling Canvas. First, the node is created on the Modeling Canvas, the 'type'-, 'id'-, and 'static'-attributes are set and the new node is connected to the 'Widget'-node. If the button is placed in a container, an additional edge is created between the 'HTML Element'-node representing the container and the new node that represents the button. Furthermore, it is possible to edit the wireframe model through the frontend component model view. For example one can create any UI control element in the Wireframing Editor though the Modeling Canvas by creating an 'HTML Element'-node, connect it to the 'Widget'-node and set the 'static'-attribute to true. After each action on the Wireframing Editor, an auto layout algorithm for directed graphs is applied to the Modeling Canvas, only manipulating those elements that were updated.

## 4.5 Evaluation

We evaluated the CAE in several evaluations, according to the our methodology as presented in Sec. 4.1. In this section, we describe each evaluation step in detail.

### 4.5.1 Initial Evaluation

We successfully used the CAE to redesign an existing collaborative Web application used for graph-based storytelling [LND\*16]. While this evaluation was only conducted internally, we used it as a first proof-of-concept usage scenario for the CAE<sup>6</sup>. It provided initial feedback on the usability and acted as a first and ongoing test case that lead to several necessary improvements of the framework, before we were able to apply it in the following user evaluations.

---

<sup>6</sup><https://github.com/wth-acis/CAE-Example-Application>

### 4.5.2 Evaluation with Heterogeneous Teams

After we successfully defined the agile and cyclic development approach that builds the basis for development with the CAE, we conducted our first user evaluation [LNKJ17].

**Participants and Procedure** We considered groups of two to three people with various technical backgrounds. We carried out 13 sessions, with a total number of 36 participants. The groups consisted of at least one experienced Web developer and at least one member without any technical experience in Web development, who received a description of the application to be designed. During the evaluation session, the non-technical members had to communicate the requirements to the developer team and collaboratively implement the application using the CAE. Each session lasted for about 45 minutes. The goal of this study was to assess the role of NRT collaboration for the development process, and whether our approach improves the integration of non-technical community members into the design and development of Web applications.

**Analysis and Outcomes** In general, we received high ratings from non-technical members in terms of methodology (“Understanding of separation of concerns” AVG of 3.91, SD of 0.97; “Understanding how application was built” AVG of 4.36, SD of 0.89), and developers felt they were able to implement the requirements formulated by the non-technical members (AVG of 4.64, SD of 0.49). Most non-technical members felt integrated well into the NRT development process (AVG of 4.27, SD of 0.98) and the oral interviews revealed that they could follow the development process well. Although the question, if non-technical members took an active role in the development process received the lowest score, the result is still pretty high (AVG of 3.82, SD of 0.96). From the developer survey, we received the highest ratings for questions regarding the concept of CAE and its usability (“Understanding functionality” AVG of 4.79, SD of 0.43; “Understanding separation of concerns” AVG of 4.71, SD of 0.83). Collaborative aspects were also rated rather high by both groups. The oral interviews revealed that most developers felt both the need for requirement analysis improvements regarding the inclusion of non-technical stakeholders as well as that the CAE can be used for this purpose.

The evaluation showed the usefulness of the CAE to integrate non-technical members better into the development process. Developers saw the benefit of CAE’s MDWE approach to contribute to a unified community application landscape. A particularly often requested feature was the introduction of a second abstraction

tier for the frontend component view, which could hide too technical aspects from non-technical members, concentrating more on the “visible” elements, putting the functionality into a second component view, which would then be used by the developers only. Another issue mentioned by the developers was the need to adjust the generated source code to fully reflect the requirements, and then having no possibility to return to the modeling environment, since the modified code would be overwritten when the code was regenerated by the framework. We used this feedback to start developing the Live Code Editor, as well as the Wireframing Editor, which tackle these issues from different angles, but both use the same idea of providing different views on the same model.

### 4.5.3 Evaluation in a Lab Course

While we were developing the aforementioned live coding and wireframing extensions, we in parallel started to validate our MDWE process with its modeling and development phases over a longer period of time, by studying the impact it has on Web developers [LNKK16, LNKJ17]. Therefore, it was necessary to extend the CAE with automated deployment features, such that the created applications could be used in practice, to validate their functionality.

**Participants and Procedure** We evaluated our approach in a lab course of 15 undergraduate computer science students. The students had basic programming knowledge, in particular in Java (AVG of 4.60, SD of 0.63) from their first programming lectures, but our pre-survey also indicated that none of them were really familiar with Web development (AVG of 1.67, SD of 1.29) or microservice architectures (AVG of 1.73, SD of 0.95). During a two week period, the students were asked to model and deploy the basic framework of their lab course prototype.

**Analysis and Outcomes** In this evaluation, we were especially interested in how the CAE can help developers that are not yet familiar with the present development environment. Our questionnaire thus focused on the learning effects the CAE has on developers that have to integrate into a new development process. Our results indicate a high learning effect in terms of understanding the underlying Web development concepts of microservices (AVG of 4.43 vs 1.73, SD of 0.65 vs 0.95) and frontend components (AVG of 4.50 vs 2.60, SD of 0.52 vs 1.45). We received rather high ratings in terms of MDWE easing the learning of new

concepts and techniques (AVG of 3.86, SD of 0.77) and MDWE improving the understanding of the generated application (AVG of 3.71, SD of 0.99).

Occurring problems during this evaluation were mainly due to the use of an experimental prototype which was never tested in an environment with more than a handful of people using it at the same time. Boundary conditions and network latency problems lead to a cycle of fixes, version incompatibilities and newly introduced problems. Even though this might have clouded the participants' impression of CAE use, it finally lead to major technical improvements of our framework. These first results of a more realistic usage setting showed promising applications of the CAE as a tool to teach developers of different domains the development of Web applications with a P2P microservice architecture.

#### 4.5.4 Live Code Editor Evaluation

Due to the feedback received in the previous two evaluations, we developed the Live Code Editor. On this, we performed a usability study with student developers to assess how it integrates into our collaborative MDWE methodology and how well it performs and is received in practice [LNWK18].

**Participants and Procedure** We carried out eight user evaluation sessions with two participants each. After receiving a short introduction and filling out a pre-survey to assess their experiences in Web development, the participants were seated in the same room and asked to extend an existing application, which consisted of two frontend components and two corresponding microservices. As expected, the pre-survey rating of the familiarity with Web technologies (AVG of 4.00, SD of 0.48) was rather high. However, only a minority of our participants were familiar with MDWE (AVG of 2.67, SD of 0.37) or had used collaborative coding for creating Web applications before (AVG of 2.40, SD of 0.69). Each evaluation session took about 30 minutes of development time. Afterwards we asked the participants to respond to a questionnaire consisting of questions on a 5-point Likert scale.

**Analysis and Outcomes** The participants rated connections between our two collaborative phases, namely the access to the code editor from the model (AVG of 4.67, SD of 0.32) and the reverse process with the synchronization enabled (AVG of 4.40, SD of 0.25) very high. Even though the chosen application was, due to the time constraints of a live evaluation setting, quite simple, the evaluation

participants mostly saw cyclic development in general as relevant (AVG of 4.13, SD of 0.43) and also rated the benefits of a cyclic MDWE process high (AVG of 4.00, SD of 0.44). Moreover, all participants identified the advantages of code and model synchronization (AVG of 4.33, SD of 0.37). We considered the results of this evaluation as a proof-of-concept, that the Live Code Editor technically fulfills its purpose as a way to integrate live coding into the cyclic development, mitigating the need to manually change the generated source code and thereby break the MDWE cycle.

### 4.5.5 Wireframing User Evaluation

After we developed and integrated the Live Code Editor, we tackled the feedback gained in our evaluation with heterogeneous teams (see Sec. 4.5.2) and developed the Wireframing Editor. We then evaluated it to gain user feedback on how well it integrates into the process and how it changes the way applications are developed with the CAE [LNRK19].

**Participants and Procedure** We recruited eight student developers as participants, which were split up into groups of two, resulting in four evaluation sessions that each lasted about 60 minutes. As in the Live Code Editor evaluation, the pre-survey revealed a high familiarity with Web development (AVG of 4.50, SD of 0.53). Also, a rather high familiarity with MDWE concepts was observed (AVG of 3.13, SD of 1.13), but wireframing editors were not very familiar to the participants (AVG of 2.75, SD of 1.58). The participants were asked to develop a frontend for an already existing microservice backend. A specification for both the existing RESTful API, as well as the desired Web frontend was handed out to the participants at the beginning of the session.

**Analysis and Outcomes** Fig. 4.13 depicts the most interesting results of our evaluation. With an average of 4.00 (SD of 2.62), most participants found the wireframing editor was easy to use and with an average of 4.13 (SD of 1.98) and 4.25 (SD of 1.41), the participants found both their application reflected in the live preview widget as it was designed by them, as well as that they were aware of what their collaborator did in the wireframing editor. Also, participants mostly agreed that the modeling canvas and the wireframing view reflected the same model (AVG of 4.25, SD of 2.33). With an average rating of 4.25 (SD of 1.77), the majority of the participants thought the wireframing editor a useful extension for MDWE



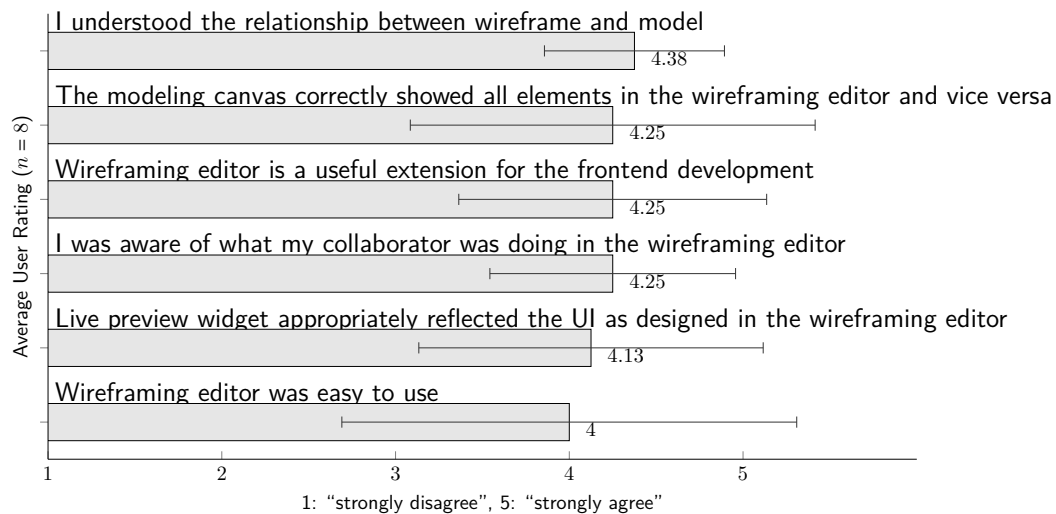


Figure 4.13: Results of the wireframing user evaluation.

frontend development and the integration of the wireframe into the process was understood quite well (AVG of 4.38, SD of 1.04).

#### 4.5.6 Wireframing Activity Evaluation

To gain a deeper understanding about the collaboration process and working behavior of the participants, we monitored the participants' activities in the CAE.

**Participants and Procedure** We monitored five sessions with two participants each. Each time a participant switched the widget, the time spend in the widget was logged. Furthermore, if a participant altered the wireframe, model or code, an additional event was logged. For each participant the time spend in each editor was aggregated and the total amount of altering activities a participant issued was counted.

**Analysis and Outcomes** Tab. 4.1 depicts for each participant the relative time spend in a certain widget, as well as the number of absolute activities in this widget. A particular activity can be a create-, move-, resize-, delete- or attribute change event of an element in the Wireframing Editor or Modeling Canvas, or a value change activity in the Live Code Editor. These results show that the participants spend the most time in the Modeling Canvas. One explanation for

|           | Participant 1 |            |           | Participant 2 |            |           |
|-----------|---------------|------------|-----------|---------------|------------|-----------|
|           | Wireframe     | Model      | Code      | Wireframe     | Model      | Code      |
| Session 1 | 20.1%, 20     | 26.1%, 180 | 53.1%, 14 | 38.9%, 18     | 27.4%, 87  | 33.7%, 2  |
| Session 2 | 7%, 3         | 81.3%, 104 | 11.7%, 4  | 15.4%, 10     | 41.7%, 133 | 42.9%, 15 |
| Session 3 | 7.1%, 12      | 79.6%, 291 | 13.4%, 20 | 17.6%, 3      | 65%, 288   | 17.4%, 6  |
| Session 4 | 11.9%, 0      | 51.5%, 169 | 36.6%, 11 | 9.4%, 8       | 57.7%, 116 | 32.9%, 5  |
| Session 5 | 53.8%, 20     | 44.6%, 61  | 1.6%, 0   | 34.8%, 19     | 56.1%, 78  | 9.1%, 0   |

Table 4.1: Results of the wireframe activity evaluation for each session and participant. The relative time spend in each widget, as well as the absolute number of activities for each widget is given.

this might be that the participants had to get familiar with the modeling language first, which corresponds with our observations, that during the first few minutes participants did not use it productively, but experimented with different modeling elements, until they were familiar with them. With an average of  $AVG_{model} = 188$  activities, the modeling part was also the most demanding task. With an average of  $AVG_{code} = 13$  activities the coding task was less work intensive. One reason for that might have been that all participants had some development experiences. The average number of activities to complete the wireframing task was quite low with  $AVG_{wireframe} = 10$ , and also the time spend in the Wireframing Editor was quite short, compared to the time spend in the Modeling Canvas. The participants of the fifth session were not able to generate the code, which is why there is almost no activity and usage time recorded in the Live Code Editor.

These results indicate that the Wireframing Editor was easy to use and required little time to get familiar with, which was a major goal and a key requirement of the editor. Nevertheless it has to be mentioned, that the evaluation of time spend in each editor and activity monitoring is tightly coupled with the evaluation task, and thus influenced by it.

#### 4.5.7 Service Success Measurement Evaluation

Our service success measurement evaluation concerned the integration of the CAE in the overarching las2peer methodology, with a special focus on the Requirements Bazaar integration and the usage of service success measurement modeling elements, which then were used for visualization in las2peer's monitoring and evaluation suite.

**Participants and Procedure** We recruited thirteen participants from the computer science department and conducted thirteen evaluation sessions. Participants were given an existing Web service for image uploading, which had a certain, yet obvious, flaw. It contained an (artificial) delay in the upload process. The service itself was already developed with the CAE and a corresponding Requirements Bazaar project existed, that already documented the flaw. The workflow of the evaluation contained first reading the documentation of the issue in the Requirements Bazaar, and then reacting by first measuring the image uploading time via a modeled monitoring message extension and finally correcting it in the Live Code Editor. The resulting improvement could then also visually be confirmed in the NRT Evaluation Center.

**Analysis and Outcomes** While the evaluation was concerned with several aspects of the NRT Evaluation Center, three questions were posed to specifically justify the Requirements Bazaar and monitoring integration into the CAE. With an average score of 4.32, most participants found the Requirements Bazaar well integrated into the CAE. Our observation was, that participants had no problems browsing the requirements connected to the modeling project directly from the CAE's interface. Another question to verify this was, if participants were able to distinguish the responsibilities of the CAE, the Requirements Bazaar and the NRT Evaluation Center. This was answered with an average score of 3.83, which, taking into account the short time the participants had to get familiar with the concept, is a clear sign that the responsibilities of the individual components were understood. Finally, we asked, if the combination of the three components was perceived as a way to make monitoring features more transparent to non-technical stakeholders, which was answered with an average score of 3.71. Taking these results together, we perceive the NRT Evaluation Center well integrated into the CAE.

#### 4.5.8 Project Management Evaluation

Finally, we evaluated the general usability and usefulness of the versioning system and the project management layer of the CAE. Therefore, we designed a Web application which then was collaboratively extended by the participants during the evaluation sessions.

**Participants and Procedure** The evaluation comprised seven sessions with two to three participants. We recruited both participants with a background in Web

development, as well as participants without any prior knowledge in this domain, which our demographic questions showed (“Web development experience using HTML and JavaScript” was answered with an average of 3 (SD of 1.71). In total we had 16 participants, and except of two sessions which were held in presence, all sessions were held online. We took care that the conditions for both the online sessions and the presence sessions were comparable. After a short introduction into the tasks and the Web application to be developed, the participants used the given task sheet to start the development of the planned application. The goal was to model a frontend component that allows to display a list of song titles after a button gets clicked, making use of an already existing microservice backend. To evaluate the versioning functionalities for viewing previous versions of a component and for comparing different versions, an update of the backend microservice was then simulated. The participants were asked to compare both versions and to identify how the extensions of the music service could be used to improve their own application. After the evaluation, participants were handed out a questionnaire which used a five-point Likert scale.

**Analysis and Outcomes** The evaluation showed that the way of how the invitations to projects work was familiar to the users. The corresponding statement was rated with an average of 4.63 (SD of 0.50). The question regarding the understanding, that projects bundle frontend components and microservices of a single Web application reached an average of 4.38 (SD of 0.72), while the question on the understanding of what dependencies are used for was rated with an average of 4.69 (SD of 0.60). This shows that the terminology used in the project management was easy to understand for the users.

The next category of questions focused on the versioning system. With an average of 4.38 (SD of 0.72) the participants agreed that the combination of the versioning widget together with the models displayed in the canvas helped to see the differences between two service versions. Very similarly rated was the statement if the users could identify how the model of the music service developed between the two versions (AVG of 4.38, SD of 0.81). Most of the groups were able to identify both differences between the two versions without additional help. In general, the participants saw the advantage of using a versioning system in a Web-based collaborative tool for building Web applications (AVG of 4.88, SD of 0.34). Besides that, with an average of 4.75 (SD of 0.45) the participants agreed that a versioning system can help to support the awareness of changes done to the model by other users.

For the evaluation of the general usability of the extension, the System Usability Scale (SUS) [Broo96] was used. It aims at measuring the subjective assessment of usability by taking effectiveness, efficiency, and satisfaction into account. We mention here the most interesting results. The statement “I found the project management or the versioning system of the CAE unnecessarily complex” received an average of 1.50 (SD of 0.63), indicating the ease of understanding the project management view of the CAE. The next statement “I thought the project management and versioning system of the CAE were easy to use” was rated with an average of 4.38 (SD of 0.72), which showed that the users found the CAE extensions easy to use. The statement “I found the various functions in the project management and versioning system of the CAE were well integrated” was rated with an average of 4.50 (SD of 0.63). The participants also agreed, that users could get familiar with the extensions quickly. The statement “I would imagine that most people would learn to use the project management and versioning system of the CAE very quickly” got an average of 4.56 (SD of 0.63). The third statement “I found the project management or versioning system of the CAE very cumbersome/difficult to use” was rated with an average of 1.38 (SD of 0.50) and the statement “I felt very confident using the project management and versioning system of the CAE” received an average of 4.31 (SD of 0.70). The final statement of the second part “I needed to learn a lot of things before I could get going with the project management and versioning system of the CAE” was rated with an average of 2.25 (SD of 1.24). All in all, we interpret these results as a sign that the project management layer is well integrated into the CAE and serves its purpose of structuring the development process.

## 4.6 SBF: The Social Bot Framework

Nowadays, virtual personal assistants are omnipresent. “General personal assistants” like Alexa, Siri and Google Assistant are used for simple tasks like providing the weather report, as well as personalized content, as for example the next scheduled appointment [YCCI16, DLTM18]. People often communicate with these personal assistants on a daily basis and their usage and thus their assistance is deeply rooted in many daily lives.

Besides these general assistants, there are also specialized versions tailored to solve domain-specific tasks. The vast majority of these are domain-focused chatbots [DALE16]. These technologies are widespread, as Facebook for example provides an own API to create chatbots, mostly used by celebrities or companies,

that can answer common questions automatically [FVD\*16]. Most of these chatbots use predefined rules to answer user inquiries and are called “retrieval-based” bots [BLer17]. In contrast, bots that are capable of adapting to the user and, for example, understand natural language are called “generative” bots [BLer17]. Generative content depends on the current context and takes into account the recent history between the bot and the user. Utilizing machine learning techniques, these bots learn from this history and can answer arbitrary questions based on it [SSG\*17, VaPi18]. Generative bots are not as widespread as retrieval-based ones, due to their higher technical complexity. In fact, in 2018, less than a third of Facebook’s most popular chatbots used natural language processing [PeDi18].

### 4.6.1 Motivation

Developing a bot for a particular Web application that directly collaborates with the user is a task that requires deep technical understanding of both the application and bot development. Requirements, that especially CoPs often do not fulfill. But the utilization of social bots in these communities provides a level of support that is, due to the lack of available resources, otherwise often not feasible to provide [RANR03].

Following the core ideas of the CAE, by using MDD for developing social bots, we can raise the abstraction level of development and thereby allow a deeper integration of end users into the bot creation process. Using a similar Web-based model-driven environment for social bots, we aim at providing CoPs with the means to create their own learning assistants, much like the CAE provides CoPs with the means to create their own Web applications. With the *Social Bot Framework*, we present a domain-independent framework, which utilizes RESTful APIs with OpenAPI specifications to define the social bot actions within an application<sup>7</sup>. CoPs can use the SBF to create their own social bots for their own (self-hosted and/or decentralized) CIS.

### 4.6.2 Concept: A Metamodel for Social Bots

Analog to the CAE, the SBF is based on a well-defined metamodel. To introduce this, Fig. 4.14 shows an example model of a social bot, used in one of our evaluations. The central entity of a social bot model is the *VLE Instance* element,

---

<sup>7</sup>The prototype implementation was partially supported by a master thesis [Neum18] under supervision of the author.

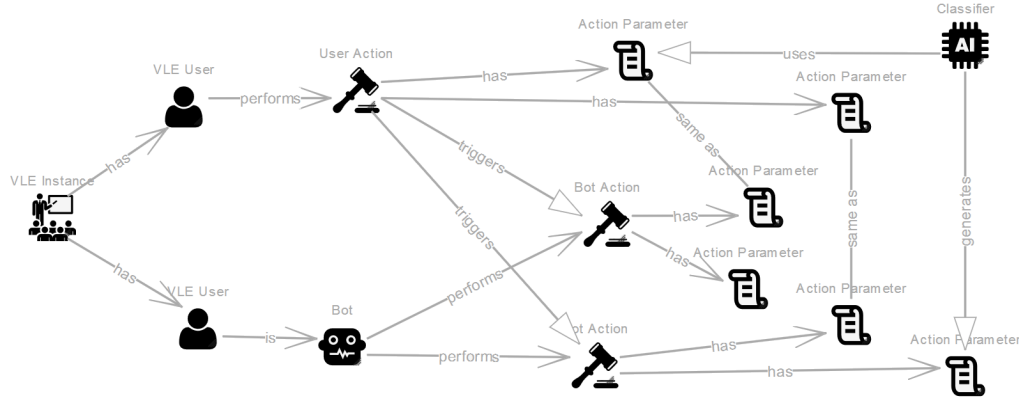


Figure 4.14: Screenshot of a social bot model.

which represents the learning environment (the application) the bot should act in. *VLE Users* act within this *VLE Instance*. These *VLE Users* are either human users or *Bots*. Hence, there exists a difference between *User Actions* and *Bot Actions*, which can be performed by these acting entities. A *User Action* refers to an action within the service. It is specified by the action name (e.g., the function call of that service). This can lead to the trigger of a *Bot Action*. These actions can both be actions of the bot within the service (analog to a user action, but performed by the bot), but they can also lead to the sending of a message via a conversational interface. *Bot-* and *User Actions* have *Action Parameters*. These relate either to typical arguments of a service call like method names and types, or in the case of a *Bot Action*, they can contain the message that should be returned via the conversational interface. *Bot Actions* are initiated by a trigger, which represents the instructional rules from the VLE and can be a *VLE Routine* or a *User Action*. A *VLE Routine* is a time-controlled trigger (e.g., a workday or a specific time). If no static value has been defined, the content is either taken over directly or can be manipulated using *If-Then Statements* or one of the two content generators *Classifier* and *TextToText*, which are utilizing deep learning technologies. With *If-Then Statements*, the retrieval-based approach is pursued, whereby content generators follow the generative approach. The latter technology requires training data for their deep learning model, which can be specified after the initialization of the bot.

The concrete example of Fig. 4.14 depicts the a bot model used in our evaluation of the SBF (cf. Sec. 4.6.4). It utilizes the Distributed Noracle application, and built the basis for the *Noracle Bot*, which we evaluated within our real-world

application of a knowledge-building infrastructure as well (cf. Sec. 5.2.4). This bot can rate questions, based on their fitting to a certain category, as well as sending users notifications of interesting questions. Here, the *VLE Instance* holds the deployment endpoint URL of the Noracle service, and the bot is modeled as a user of the service. The *User Action* holds the function to create a question and requires two parameters: the corresponding learning space and the content of the question. Creating a question triggers two bot actions. The first evaluates the created question and the second sends a notification to a specified channel of a conversational interface. The rating of a question requires as parameter a score, which is defined by the classifier. The notification via a conversational interface requires as parameters the respective channel and the content of the message.

### 4.6.3 Realization: Social Bot Life-Cycle

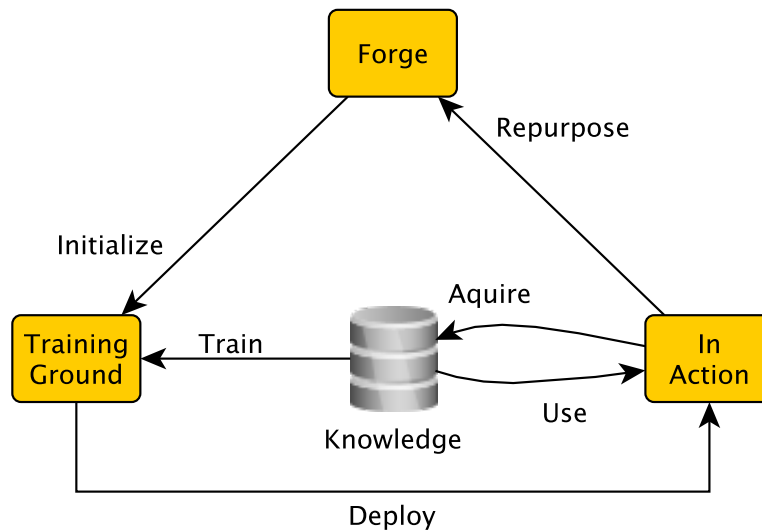


Figure 4.15: Social bot life-cycle.

This section describes the life-cycle of a social bot, which is also depicted in Fig. 4.15. A bot life-cycle evolves around its *Knowledge* (depicted at the center of the figure) and consists of three *Phases* (depicted as yellow boxes around it). The creation of the bot happens in the *Forge*, where it is equipped with its initial knowledge. This initialized bot is then sent to the *Training Ground*, where it is



consequently trained, based on the knowledge database. When the bot's training is done, it is send into *Action*. Here, the bot acquires further knowledge through its action, which it in term uses to improve its behavior within its deployment. In the following, we describe each of the three phases of this metaphor in technical detail.

### Bot Forge

The *Forge* is derived directly from the CAE. Thus, it is a Web-based environment, based on SyncMeta, which uses Yjs to realize the NRT collaboration. It contains the same frontend UI elements for modeling as the CAE, namely the palette, the modeling canvas and the activity widget, albeit with the SBF metamodel we presented earlier on in this section. The backend is based on a las2peer service. Here, the bot model is created. During the modeling process, the collaborators specify the deployment endpoint URL of the VLE instance, and the forge fetches the available methods of the service. These methods are then displayed interactively in the modeling environment and provide the collaborators with an overview of interaction possibilities of the bot with the application (see Fig. 4.16). With this, arbitrary Web services that follow the OpenAPI specification can be used by the bot to interact with. Integration of community tools like Discourse<sup>8</sup> or LMS like Blackboard<sup>9</sup> is thus possible.

As a conceptual member of the CoP, the bot then can be told to interact with the learning service by using the modeling element *Bot Action*, to act within a learning environment as other (human) members would. The actions of a social bot are initiated by modeled *Triggers* (see also Fig. 4.14). To avoid endless loops, a bot is not allowed to respond to its own actions. The result of this modeling process is a bot in the state of being “initialized”.

### Bot Training Ground

When it comes to advanced queries, retrieval-based bots reach their limits. Therefore, we utilize the open source deep learning technology TensorFlow [ABC\*16] to create generative content. This happens in the *Bot Training Ground*. The prerequisite for this training is a social bot that has been initialized so that one can create a *Knowledge Model* for it. The use of deep learning elements is suitable for adequate actions, but they must be fed with training data to provide appropriate

---

<sup>8</sup><https://docs.discourse.org/swagger.json>

<sup>9</sup><https://developer.blackboard.com/portal/displayApi>

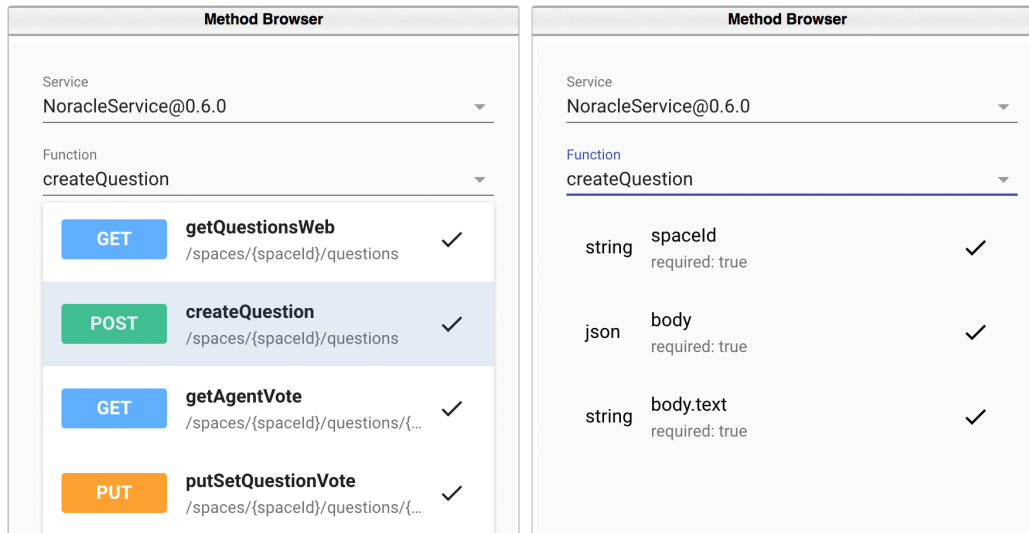


Figure 4.16: Screenshot of the method browser, in this example depicting the available methods of the Distributed Noracle service.

results. In order to turn the information from a *las2peer* service into training data sets, we hook into the MobSOS monitoring pipeline. By extracting service specific data and linking it in a coherent way, the training data is generated.

To give an example, when evaluating a text, this text can be seen as input and the evaluation as expected output. This results in a *String*-to-*Int* relation which can be forwarded to a text classifier [Kim14]. Text generation, for example, can use a response to a text as training data. This results in a *String*-to-*String* relation. The data is split into a training- and a validation set.

The openness of integrating with arbitrary (external) Web services enables a variety of additional possibilities. To be able to access training data from third-party providers, we assume that these materials can be retrieved with the help of REST API calls. List. 4.1 shows a query used in our evaluation that retrieves example training data for the domain of “Java” related questions. The query refers to questions from Stack Overflow<sup>10</sup> and was used with the data explorer of Stack Exchange<sup>11</sup>. The name of the programming language is replaced by an integer value to obtain the appropriate data format. After the classification, the value can be mapped back again to the respective string.

<sup>10</sup><https://stackoverflow.com>

<sup>11</sup><https://data.stackexchange.com/stackoverflow>

Listing 4.1: Query for the 10,000 most upvoted Java questions on Stack Overflow.

```
select top 10000 p.title , 'java'
from votes v
inner join posts p on p.id=v.postid
inner join PostTags on PostTags.Postid=v.postid
inner join Tags on Tags.id=PostTags.Tagid
where PostTypeId = 1 and Tags like '%<java>%'
group by v.postid,p.title, p.tags
order by count(v.postid) desc
```

A simple classifier can then, for example, decide whether Java questions are involved or not, by mapping the query from List. 4.1 to “1” and setting other programming languages to “0”. These training data sets are then forwarded to the open source deep-learning technology. Currently, we use both a text classifier<sup>12</sup> and a text generator<sup>13</sup>, but there are various other possibilities (e.g, image to text) that could be explored in the future as well.

Fig. 4.17 shows the Web interface of the Bot Forge. It allows the user to browse available generative elements of the previously modeled bot. The status, whether the element has to be trained, is in training, or has finished the training, can be fetched with the help of the “Check” button. Starting the training initiates a Python instance where the deep learning model acquires knowledge from the specified training set. The number of training units depends on the respective training set and the deep-learning technology. These steps require some kind of expertise, so for training a bot with deep learning techniques on “newly generated” training sets, domain experts or data scientists are required. For available training sets, or previously trained bots from a similar domain, users can select to train the bot the same way by clicking the corresponding button.

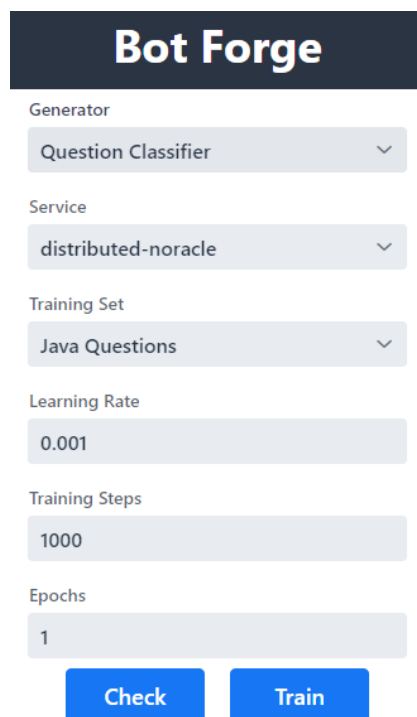
### Bot in Action

Once the first training session has been completed, the bot is ready to be deployed and set to action. In order to be able to interact with the CoP, the social bot must register in the learning space it was developed for. This is done, for example, by the community who adds the bot to a learning room. The bot then waits for a trigger as described in Sec. 4.6.3. Every action of a social bot is also tracked by the monitoring pipeline and can be used in combination with incoming user data for continuous training of the deep learning model. The bot can interact

---

<sup>12</sup><https://github.com/dennybritz/cnn-text-classification-tf>

<sup>13</sup><https://github.com/karpathy/char-rnn>



The screenshot shows the 'Bot Forge' web interface. It features a dark blue header with the title 'Bot Forge' in white. Below the header, there are several configuration options, each with a label and a corresponding input field or dropdown menu. The options are: 'Generator' with a dropdown menu showing 'Question Classifier'; 'Service' with a dropdown menu showing 'distributed-noracle'; 'Training Set' with a dropdown menu showing 'Java Questions'; 'Learning Rate' with a text input field showing '0.001'; 'Training Steps' with a text input field showing '1000'; and 'Epochs' with a text input field showing '1'. At the bottom of the form, there are two blue buttons: 'Check' and 'Train'.

Figure 4.17: Screenshot of the Bot Forge Web interface.

within the VLE and also send notifications in external conversational interfaces. The framework attempts to automatically match the VLE user with the user of the external communication channel. A prerequisite for this to work is either the same username or email address. It is thus possible to send messages to a predefined channel (group of users) or to a specific user. The communication channel implemented within the SBF is modular and offers an analog integration of other messengers. In its current state it integrates with Slack and Rocket.Chat.

#### 4.6.4 Evaluation

A user evaluation gave a group of participants the opportunity to create a simple social bot, that interacted with a learning service. As already mentioned in the previous sections, we used the *Distributed Noracle* application (cf. Sec. 5.2) as evaluation scenario.

**Participants and Procedure** The evaluation comprised six sessions with two participants per session. Thus, it involved 12 users, recruited from members and students of university departments in the fields of medicine, business administration and computer science. The participants were asked to load a template of a bot model, adjust the parameters and extend it. Each session was conducted locally, with participants being in the same room together with one facilitator, and the participants were allowed to talk to each other during the modeling process. A brief introduction to the system was given at the beginning. The task was divided into two parts, first the bot should evaluate the questions created in the space, and second notify the users in a provided conversational channel when someone reacted to a question. The necessary attributes for the conversational integration were available to the participants as an excerpt of the API. Since there was no previous learning space with enough information to evaluate the questions, we created and populated a learning space consisting of questions from programming languages (data sets were obtained as described in Sec. 4.6.3) in advance. Thus, training data sets were transferred to the monitoring system of our microservice architecture and could be selected by the users. Finally, participants were asked to complete a questionnaire. The questions were formalized as a 5-point Likert scale.

**Analysis and Outcomes** The following results refer to the outcomes of the survey and the observations made during the evaluation process. The questions are divided into two categories. For evaluating the usability of our approach we used the SUS questionnaire. The second category contained answers to free text fields and three additional questions regarding the framework's functionality. Overall, we received 11 responses. Our pre-survey revealed a moderate familiarity with RESTful Web services, with an average score of 3.80. A similar familiarity was observed for Web application development with an average rating of 3.70. The question whether users were familiar with social bot generation platforms resulted in an average of 2.10. As expected, only few mentioned that they had already created a social bot for Telegram or Facebook Messenger.

The results of the SUS questionnaire can be found in Fig. 4.18 (latter ten questions). The test persons did not belong to any common CoP using the service, so that only partial interest was addressed here. Some users were also unfamiliar with the used service and the introduction phase turned out to be a bit short. But it is noteworthy that, with the exception of one group, all groups managed to model a functioning social bot. The well-established integration and low inconsistency,

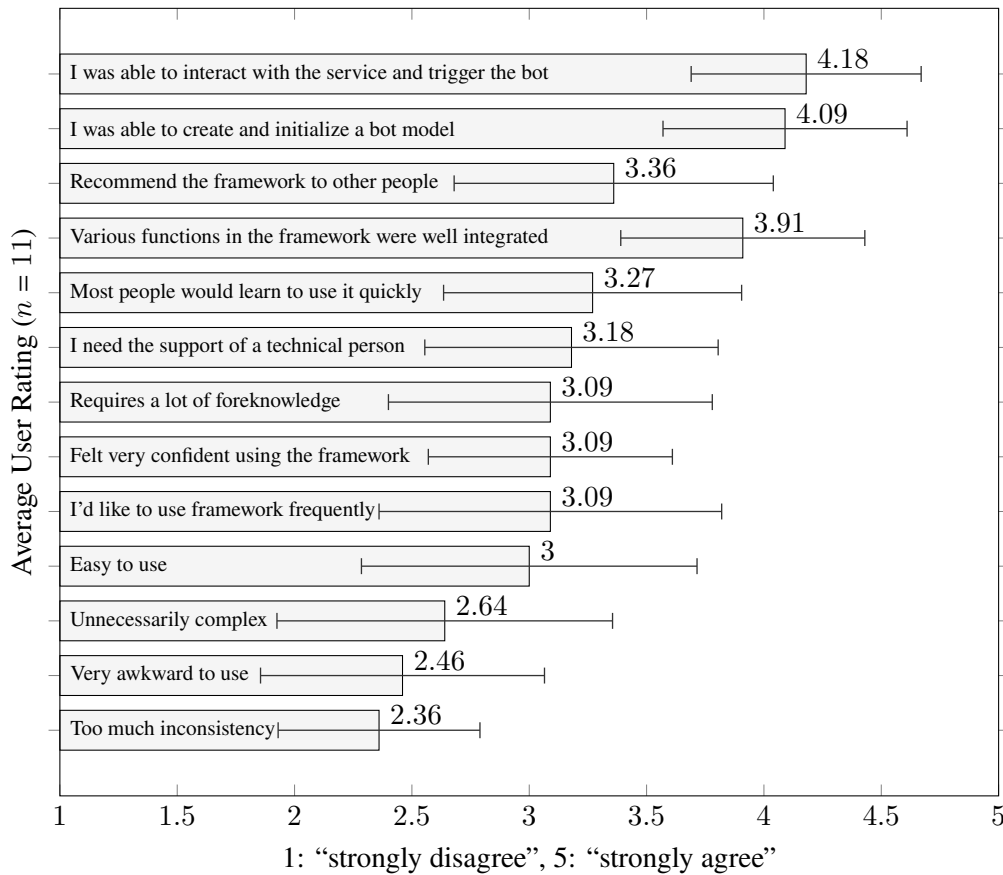


Figure 4.18: Results of the SBF user evaluation.

show that users quickly became familiar with the system and the semantics behind the model did lead to the desired outcome. There were a few users who thought the system was a bit laborious to use, but the majority accepted the complexity of the framework as manageable. This shows that our prototype is already in a usable state, although there is still room for improvement regarding the usability of the modeling environment. One group at a certain point started to try out all functionalities of the modeling tool without following a specific goal. We track these “insecure actions” back to the aforementioned short introduction. The users had a steep learning curve to get familiar with the service which the bot had to be modeled for.

In addition to the SUS statements, three additional statements were added, the first addressed whether the participants would recommend the framework to other

people or not. The final two questions relate to the success of modeling and using the social bot. The initialization and triggering in the last two questions refer to both tasks. All but one group were able to complete the first task to load and adapt the template model. Only two groups were not able to trigger the bot in the second task on their own. As an exemplary reason for these problems, one of the groups forgot the connection of a *User Action* triggering a *Bot Action*. But issues like these were fixed with the help of the evaluator already during the evaluation and the participants then acknowledged they understood the then facilitated outcome.

Our general impression during the sessions was that the users felt neither overly confident nor unsure. Users did state that they would have liked a longer introductory phase or a more detailed tutorial on the framework in order to be able to work better with it. The observation also showed that the people with a higher level of technical skill tried to understand the whole concept, whereas groups with a less technical background just used the framework without questioning its inner workings. Overall, we can state that the SBF proved to be a usable tool to create social bots for VLEs and that it is for the most parts usable by non-technical community members. Further evaluation of the process of social bot creation using the SBF is currently undertaken and also subject to future work [NLK\*20].

## 4.7 Discussion and Conclusion

In this chapter, we introduced a methodology and Web-based tool support to foster the scaffolding of decentralized CIS for CoPs. We introduced and evaluated two main artifacts, the CAE and the SBF. The CAE, with its combination of project-based service development management, requirements engineering and wireframing support, and its capabilities of service success evaluation of running services, covers the complete application life-cycle of a community application, deployed in a decentralized CIS infrastructure. The SBF on the other hand concentrates on opening up new interaction possibilities with Web-based community applications, focusing on supporting and scaffolding learning scenarios by leveraging conversational interfaces that are already well-established within the supported communities. Combining these two approaches, we provided an answer to our second overarching research question “How to support the creation of decentralized CIS with the help of Web-based tool support?” by providing different levels of scaffolding throughout different stages of the application development and life-cycle. In the upcoming chapter, we validate both the frameworks created within this and the previous chapter, by applying the methods and the tools created within

three large application and evaluations.



## Chapter 5

# Distributed Learning in Decentralized Community Information Systems

### Summary

In this chapter, we present three real-world demonstrations and evaluations of the tools and methodologies we presented in the previous two chapters. The first is a virtual VET center, created with a model-driven methodology and using las2peer's monitoring and evaluation suite. The second is a decentralized infrastructure for knowledge-building, created for an EVS training course. The final and still ongoing demonstration implements an infrastructure for scaling up digital mentoring processes.

**Contributions**  $\Rightarrow$  RQ 3. *Keywords:* TEL; VET; LA; Distributed Learning Infrastructure; Community Knowledge Building; Scalable Mentoring. The results presented here have been partially published in [LNK117, LFGK17, LGFK18, LNNK18, LGF\*20, NLKK19, KLN\*20, NLK\*20, LBNK21]. This chapter contains partially information and content extracted from these publications.

To apply and evaluate the decentralized CIS infrastructure together with its scaffolding tools and methodologies we presented until now in this dissertation, we performed three real-world demonstrations and evaluations within the domain of TEL. More specifically, we conducted a study in the domain of Vocational

Educational Training (VET), where we created a model-driven LMS to create Personal Learning Environments (PLEs) that feature both synchronous class-style learning and asynchronous, SRL. This low-level entry barrier modeling environment was used by multiple vocational training companies across Europe to create two training courses in the domains of *Social Entrepreneurship* and *Tourism & Hospitality*. Apart from applying a derived version of our model-driven methodology for end user training course creation on the Web, we also applied the NRT evaluation center to compare SRL with a synchronous, webinar-driven learning mode, both applied in the same platform. The second real-world demonstration and evaluation was performed in the domain of EVS training. We created a fully decentralized digital version of a proven method for ignorance modeling, named *question-based dialog*. The outcome, the “Distributed Noracle”, was evaluated in multiple iterations and integrated most of the infrastructure, scaffolding tools and methodologies introduced in this dissertation. Using las2peer as basis for application development, we also used the CAE to create a first scaffold for an initial prototype of the Distributed Noracle. In subsequent iterations, we integrated both the service explorer, the NRT evaluation center and the SBF into the application. Requirements for developing and improving various parts of this dissertation stemmed largely from the iterative evaluations, as was the development of the Distributed Noracle in terms influenced by the new availability of the tool support developed during the course of the dissertation. Finally, our third demonstration presents a distributed architecture for scaling up mentoring processes. This still ongoing effort is applied in the scope of a large German research project and integrates las2peer into a kubernetes-based infrastructure, together with various applications developed by other partners of the consortium. It combines and opens up the las2peer ecosystem with external tools, dependencies and (development) methodologies by integrating it in a large ecosystem of an heterogeneous application landscape. Here, the blockchain-based verification of LA, as well as the SBF are applied at a large scale. The testbed-based structure of the project allows for large evaluation setups with hundreds of participants, of which much is still ongoing works. Still, the initial steps were made in the course of this dissertation and lay the groundwork for future promising research outcomes in the domain of decentralized and distributed scaling of (university) mentoring processes.

## 5.1 A Virtual Vocational Training Center

In this section, we present the development and evaluation of the Virtus Virtual VET Center. Although we refrain to present a complete design-science methodology for it, we first presented the center as a demonstration paper in [LNK117]. The feedback gathered from this demonstration, together with the feedback we received from the project partners and evaluation participants led us to developing and tailoring MobSOS and the NRT evaluation center more closely to the created PLEs. While doing this, and after our first evaluations also presented it more clearly, we identified the presence of two very different learning modes within the same platform, something we deemed pretty unique to our use case. Thus, we investigated this topic by comparing the learning modes side by side, using (and further extending) the monitoring capabilities of the underlying platform. The results of this study were published in [LNNK18]. The combined results of these works make up the complete demonstration and evaluation of the V3C that is presented here.

### 5.1.1 Introduction

The digital transformation is affecting vocational training as any other business these days. A virtual vocational training center is a hosted solution for the implementation of training courses in the form of for example webinars. There are many undeniable advantages coming with the utilization of virtual training centers, both for training providers and trainees. Training providers can save the rent for training facilities including the procurement of furniture, media technology and training materials. Furthermore, they can expand their business to other regions. Scaling their business is important for many training providers, in case their business is threatened by economic up- and downturns or by changing demands in job descriptions. They can enter other markets, e.g., translate electronic training materials, adjust materials to different job descriptions or cooperate with other training providers in larger virtual training centers at low front-off costs. Trainees can save travel time and costs and take parts in training programs offered outside their living areas. However, the virtualization of training has also downsides, most important the loss of social interaction with other learners and the informal exchange of information before, during and after the training. So, the goal of a virtual training center is not only the formal implementation of training courses with the necessary training, testing and certification procedures but also a re-establishment of informal and social learning opportunities.

A state-of-the-art virtual vocational training center has to combine a LMS with a PLE [Harm06]. The LMS has to be designed to be usable by all stakeholders responsible for course (content) creation, thus it has to have a low technical entry barrier. The PLE should allow for both synchronous, class-style learning and asynchronous SRL at the same time, to cope with the requirements that vocational training brings. Whilst both the need for classical teaching via webinars is needed in this domain, vocational training needs to take into account asynchronous learning, since many students are working full-time at different companies, only connected to each other as a CoP, making it impossible for them to participate in each teaching session. A platform that supports both learning modes needs to have the possibility for a live video chat, as well as for static content that is available all the time, like slidesets, videos and online assessment tools that provide feedback to both learners and tutors about the students' progress. Additionally, the system should track the learning progress of the students in form of data collection in the background. This data and its aggregation, clustering and evaluation allows for LA, which in term can provide valuable insights for both trainees and training providers to either improve their learning or teaching (material) accordingly.

As the main outcome of the European Erasmus+ project "VIRTUS", we therefore developed the Virtus Virtual VET Center (V3C). It offers webinar-style training courses with the facilitation of trainers in a synchronous manner as well as self-regulated, asynchronous learning spaces, where learners can learn in a self-paced, socially-aware way together with other learners or alone.

One main goal of this study was to apply and tailor the model-driven methodology for a Web-based editor that creates vocational training courses supporting both synchronous webinar-style (or MOOC-style) learning and informal SRL in vocational training. The resulting PLE, generated from the model-driven creation process, is realized as a flexible Web-based platform that fulfills the business goals of vocational training centers as well as learning goals of small- and medium-sized companies.

From a TEL perspective, we were interested in measuring the impact of the different learning modes on the learning outcomes. We developed and applied several LA methods, based on the principles of the NRT evaluation center we presented in the previous chapter. These methods are utilized and combined in a way to get insights into both synchronous and asynchronous learning modes, using different data collection methods based on the different interaction possibilities and also different data analysis and visualization procedures, combining them to an integrated holistic LA approach for vocational training centers. This blending of learning modes makes it possible to track the learners' progress throughout the

different learning phases and enables both learners and tutors to see differences and similarities in the learning outcomes, thereby reflecting on the learning progress.

We evaluate our approach with a focus on the comparison of the two learning modes with a large number of participants in two training courses. These training courses, created by specialist companies of different European nations, are European Credit System for Vocational Education and Training (ECVET) certified and translated into several language, used by an international audience of vocational training attendees.

### 5.1.2 Use Case: Educational Vocational Training

To give an overview on the usage scenario the V3C fulfills, we here sketch a use case that depicts a typical situation it is used in. As owner of the Gourmet Travel Agency (GTA), Chris, a V3C customer, needs to train his employees in using digital technology to book complete packages online. He therefore navigates to the V3C platform and finds a basic travel agent course offered by the V3C provider Traveling Agents Training Services (TATS), including the option of a later ECVET certification. Based on his needs, Tanja, a trainer at TATS, negotiates a customization of the course to his particular business needs. She offers to reuse a basic course by adding GTA-specific metadata to the V3C platform which contains three modules: “flight booking”, “hotel booking” and “rental car booking”, each of them featuring both synchronous and asynchronous elements. Synchronous elements involve NRT communication and collaboration technology and moderation by a designated trainer at TATS. Asynchronous elements involve different kinds of multimedia content delivery elements for knowledge acquisition: a slide presenter, a document viewer, a data upload widget and a video player. The course furthermore offers intermediate assessments after each module, as well as a final assessment at the end of the course. Chris asks for an additional course module for training gourmet restaurant booking, including information on Michelin categories, and books a full course.

Together with the TATS team of professional trainers, Tanja designs this new course module, including curriculum, multimedia contents, custom interactive course elements and the quizzes for each unit. The elements are added by dragging and dropping existing widgets to the course units. Quizzes are developed based on the targeted skills and the requirements of the ECVET certification. For Chris’ several international employees, English and German are added as supported languages. Upon completion of the new custom gourmet module, Tanja creates/exports the course using the V3C platform into the PLE. Each unit is

available as a separate activity under the course space address, the widgets under each activity reflecting the design view from the LMS component. Chris' employees at GTA navigate to the V3C platform to take the new course over a period of several weeks under Tanja's supervision. During the whole process, both Tanja and Chris' employees can view their performance analyzed by the integrated LA module. Upon course completion, learners are directed to an external certification platform for examination and official certification.

### 5.1.3 Realization

We realized our platform as a hyper learning environment consisting of a LMS and a PLE. Since the modeling environment had to fulfill a very specific use case, we decided to re-implement it from scratch instead of reusing the CAE itself. While following the same principles our scaffolding environment follows, this allowed us to create a very custom-tailored experience for training course creators in the domain of VET, which could be used without any prior Web development knowledge or introduction to it.

The LMS, consisting of both the (also) employee-facing course selection and the modeling environment is based on a stack of well-established Web development languages and protocols, such as PHP, a MariaDB database, JavaScript, and HTML5. The PLE, implemented by the courses generated by our LMS of our platform are represented as "learning spaces", realized using the Responsive Open Learning Environment (ROLE) platform [RKKN15] and also implements techniques like Extensible Messaging and Presence Protocol (XMPP) and Web Real-Time Communication (WebRTC) that realize the collaborative features like the textual and video chat platform. The monitoring was realized using an early version of the NRT evaluation center.

The LMS allows for a model-driven drag-and-drop design of course rooms. This eases the creation of courses, especially for non-technical learning designers, since it shows the learning room already in a What You See Is What You Get (WYSIWYG) fashion. To given an impression of the functionality of the modeling environment and the corresponding generated PLE, Fig. 5.1 shows an example screenshot. Here, one can see in the upper part the modeling environment. Here, there are four widgets currently added to the course room (depicted in the background) and the a editing dialog is currently opened. The lower part shows the PLE generated from this view, with the four widgets deployed within a course room.

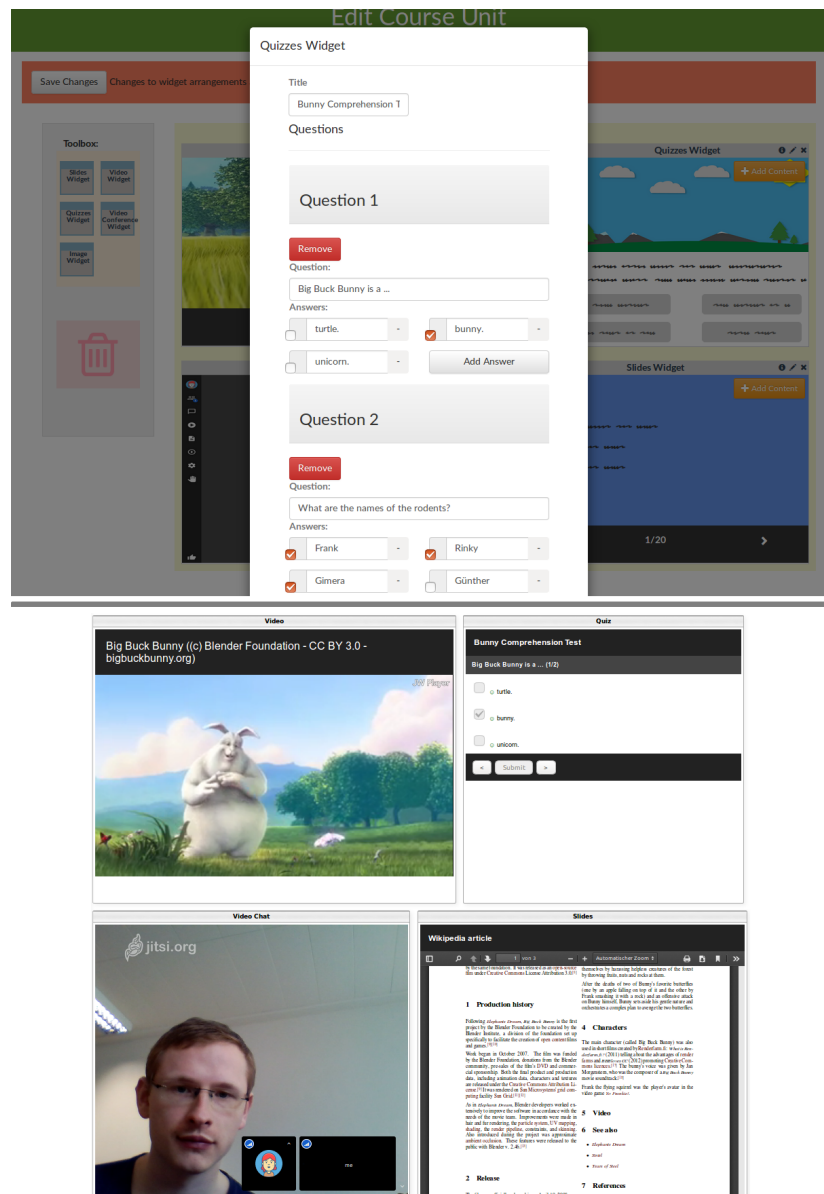


Figure 5.1: Above: The modeling view of the V3C. Below: The resulting PLE.

Fig. 5.2 shows one of the learning activities of the course “Social Entrepreneurship”. Each space represents a designed module, with its individual learning units being represented as “learning activities” of the space. Each course is divided first into several modules, which again are divided into multiple learning units. This re-



Figure 5.2: A course room of the V3C platform.

alizes a separation between the individual units which are part of a course, enables activity and progress tracking for individual units and allows for assessment via quizzes of the respective units' learning outcomes. V3C users can autonomously join spaces via the respective course unit in their LMS. Each course unit may consist of a video chat, slide presentations, various multimedia content such as audio recordings, videos and images, and self-assessment quizzes. Tutors and other learners can intervene into the learning process at any point via video or text chat, available for each course.

Since our target group for both learning designers and learners consists of people from Italy, Austria, Greece and Spain, the V3C is developed with extensive translation functionality, providing opportunities to offer and translate learning units into different languages, as can be seen in Fig. 5.3. Finally, our platform is linked to the European Certification and Qualification Association (ECQA), which then conducts and assesses the final certified exams. For data protection, we use the OIDC standard to feature a unified login for both the LMS and the PLE, as well as for the certification process at the ECQA.

The LMS of the V3C platform has a per-module LA section where tutors can see usage statistics as well as progress reports of the course participants. While learners use the PLE, usage information is logged into a MariaDB database. We track user interactions as clicks on different HTML elements. We also implemented a routine which measures the time spent on the respective module unit. All





**Social Entrepreneurship courses**

English | Search

English | Deutsch | Español | Italiano | ελληνικά | All Courses

Home | Courses | Petru PN Nicolaescu

**Course**

| Course  | Created by | Start Dates         |  |              |                   |
|---|------------|---------------------|--|--------------|-------------------|
| Social Entrepreneurship 101                           | BEST       | 2017-04-09 21:39:06 |  | Translate to | Enter Course Room |
| Description   |            |                     |  |              |                   |
| Case Study: Endeavor Greece                           | BEST       | 2017-04-09 21:41:03 |  | Translate to | Enter Course Room |
| Description   |            |                     |  |              |                   |
| Ernesto Sirolli's approach to Social Entrepreneurship | BEST       | 2017-04-09 21:49:00 |  | Translate to | Enter Course Room |
| Description   |            |                     |  |              |                   |
| Scaling the Social Enterprise                         | BEST       | 2017-04-09 21:50:01 |  | Translate to | Enter Course Room |

Add new course

Figure 5.3: The basic course edit view, showing both the multi-language feature and the translation option in the learning-designer view.

visualized information shown on the platform are real-time LA. The analytics section is split up into three different views (participants, feedback and activity). The first view contains information about individual participant results in the learning module selected. It shows the total number of participants of the module and lists them. Since during the creation progress of a module and its units, the tutor has specified the ECVET points that are later granted to students passing the final test of the course, we weight these points with a factor, resulting in a minimum duration the user has to interact with the platform to have a “full completion rate” of the course. Another information to be observed in this view is the assessment of the module. At the end of each module, the user has to complete a quiz. By clicking on a participant, the tutor can inspect the monitored data, split up into the module’s units. It also shows the previously described data of each unit, as well as how much of the quiz was completed and how many correct answers were

given. In the second view, tutors can see feedback of the learners. The feedback widget is optional for the learners and serves as a means for the learners (especially when using the platform in the asynchronous, SRL mode) to get in contact with the course designer to mention problems with, for example, understanding the content. The last view of the analytics section displays the “activity graph” of the module. All interactions tracked by the system and the overall activity of the module is displayed here. This visual analytics support enables tutors to track the activity of their modules over time, being able to see at what times students engage themselves most in the platform or which courses are more frequently visited. The graph is auto-scaled and shows the whole activity since the beginning of the module, but it also offers the option to select a desired time period. We show a screenshot of these graphs later in the evaluation in Fig. 5.4.

### 5.1.4 Evaluation

In the following, we describe the evaluation of the usage of the V3C platform. Here, we follow an established practice of data analysis [JMRy11]. Starting with a hypothesis, which shapes our research questions, we then describe the data sources and our method of analysis. We continue by presenting our results, of which the questionnaire results are based on a technical report created by the VIRTUS project consortium in [Fond17]. We close this evaluation with the interpretation of the results and limitations of our study.

#### Hypotheses

In order to evaluate our collected data and gain insightful conclusion regarding the usefulness of our approach, we formulate two research questions we want to answer with our evaluation.

**RQ1: How well does platform immanent monitoring of student behavior reflect the learning process and how well does the data collection work?** This question for one aims at evaluating the effectiveness of our implemented solution regarding its technical capabilities of dealing with learner-generated amounts of big data. Second, it aims at identifying its capability to successfully make predictions regarding future assessment results, based on the learners activity, with a special focus on at-risk students.

**RQ2: How does asynchronous, SRL perform, compared to synchronous, webinar-driven learning modes, if applied in the same platform?** Here, we want to compare the two contrary learning modes, to evaluate their differences in

final assessment results, drop-out rates and activity or engagement with the platform. As our clustering of students into synchronous and asynchronous “evaluation groups” is done at random, our question is not aiming at categorizing students into different “types of learners”, but at evaluating what effect on the learning outcomes can be seen when replacing mandatory, tutor-supported synchronous learning sessions with additional time for SRL. Since our platform allows the evaluation of the two groups in the same learning rooms, we can achieve comparable results.

### Data Sources

The data used in this evaluation was gathered from the platform immanent LA mechanism, additionally complemented with the use of a questionnaire that addressed the general impressions participants had while using the platform. Our evaluation participants were either already workers in the tourism sector or planned to enter this vocational area. Therefore, the courses offered by the platform concerned the two domains of (*Social Entrepreneurship* and *Tourism & Hospitality*). We had a total of 114 learners from four different countries, namely Austria, Italy, Greece and Spain. The evaluation period span two months. The synchronous learning phase was performed in both courses, facilitated by partners of the project. Due to practical reasons, we nevertheless only evaluated in detail the first four modules of the course *Tourism & Hospitality*, which in total covered 18 units. Every module consisted of at least two units, with a maximum of up to five units. We were able to recruit 15 learners for the synchronous and 72 for the asynchronous evaluation. Each of the four vocational training webinars spanned two hours and covered one module of the course, resulting in a four-day streak of consecutive synchronous evaluation sessions. The tutor facilitating the synchronous learning session spoke English and used the platform-immanent video conference widget. Starting with the beginning of the first synchronous evaluation session, learners who participated in this phase had two weeks time to complete the course. The asynchronous learning phase began simultaneously with the synchronous learning phase, but here learners had 60 days to complete the course. Tab. 5.1 shows the number of learners participating in each module. Finally, we invited the participants to fill out a questionnaire. We received 48 submissions here, without clustering them into the two evaluation clusters.

| Module | Synchronous<br>Participants | Asynchronous<br>Participants |
|--------|-----------------------------|------------------------------|
| SE 1   | 0                           | 42                           |
| SE 2   | 4                           | 50                           |
| SE 3   | 0                           | 37                           |
| SE 4   | 0                           | 37                           |
| SE 5   | 3                           | 34                           |
| TH 1   | 15                          | 72                           |
| TH 2   | 14                          | 58                           |
| TH 3   | 15                          | 55                           |
| TH 4   | 14                          | 54                           |
| TH 5   | 9                           | 51                           |

Table 5.1: V3C evaluation participants per module.  
(SE = Social Entrepreneurship, TH = Tourism & Hospitality)

### Method of Analysis

For collecting and public provision of our LA, we used the NRT evaluation center (cf. Sec. 4.3.4). The MobSOS Query Visualizer (MobSOS QV) is used to embed multiple measures together in a dashboard-like fashion, which we provide publicly on the platform. The stored queries for those visualizations are parameterized and can be used for every module and unit. For our analysis, we use the user activity and the scored result of the user. We define an activity as any click interaction of the learner with the PLE, e.g., a click on any button, watching a video or answering a quiz.

During the synchronous evaluation sessions, participants were allowed to use the whole platform. Although these sessions were conducted in English, participants at this time also were able to use the platform in their desired language. After each session, the tutor manually created an attendance list, which marked the attendants as part of the synchronous evaluation group, allowing us to cluster the participants into an asynchronous and a synchronous cluster.

Both asynchronous and synchronous learners used the same learning rooms simultaneously. A module is marked as completed when the participant completed the quiz at the end of it. We only consider learners who participated in each module of the course for the comparison of the scored result and the activity in the PLE. The scored result of a learner is the percentage of the correct questions of all modules. The activity is the percentage of all activities done by a user in the whole

course compared with the activities done by the learner who has been most active in the course. We base the calculation of the dropout rate by only taking those participants who finished all previous modules of a course, although our platform offers the possibility to only attend single modules of a course without the need to attend a previous module.

For further data which could not be derived by platform immanent monitoring, we used additional five-level Likert items for yet missing, however relevant subjective factors. The questions of the survey were translated into the participants native languages and were handed out in digital or printed format to the participants of both phases. Questions regard both the platform as well as the teaching material, and we focus on those that deal with user satisfaction of the system since this is part of the success awareness. We have no means to link the questionnaire to individual persons, nor do we cluster them into asynchronous and synchronous participant answers.

While using the platform, all participants were aware of the monitoring of their progress and behavior by our system, and they had the possibility to always review the personal data we collected, such as quiz results and the time they spent in a learning space. Only the overview of all participants of a module was restricted to the tutor/learning designer, as well as the visualization of the aggregated and anonymized learning room activity. This “Module Activity” view is also depicted in Fig. 5.4 in the upcoming section.

## Results

In this section, we present the results of our evaluation. It has to be noted that we only consider the first four modules of the course *Tourism & Hospitality* for analysis, since it is the only course where we have a substantial amount of participants in the synchronous learning evaluation (also cf. Tab. 5.1), which enables the comparison between the two learning modes.

As already mentioned in the previous section, our platform uses (parts of) the NRT evaluation center to implement a visual LA feature for tutors, teachers and learning designers to monitor the activity of their modules over time. Fig. 5.4 shows this view for the first four modules of the course. It displays the absolute number of activities for these modules over time. The timespan in this figure was chosen to match the sixty day evaluation span. It can be seen that two peaks in every module exist, which correspond to the final assessment of both the synchronous evaluation phase (after 14 days) and the one of the asynchronous evaluation around six weeks later. As it can also be observed, there is a smaller, yet considerably

high activity over the whole evaluation cycle, which is an indicator for continuous usage of the platform by the participants during our evaluation.

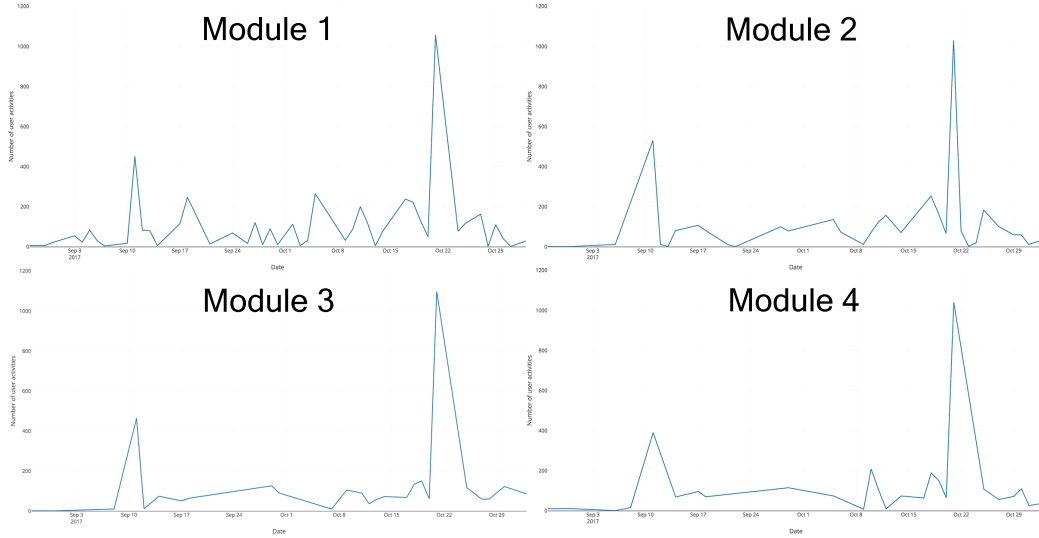


Figure 5.4: V3C evaluation: Module activity over time.

Our next analysis compares the participants' activity with their scored result in the assessment. Fig. 5.5 shows this comparison. Here, we use relative values for both activity and scored result to make them comparable, due to the different number of participants in both evaluation phases. We only consider those learners that took part in the whole course. The Pearson correlation coefficients over all four modules are  $p = 0.4593$  for the asynchronous phase and  $p = 0.3589$  for the synchronous phase.

Fig. 5.6 gives an overview about the final aggregated results of the participants for both evaluations. Again, we use relative values here to make the results comparable and clean the data from all prior drop-out participants.

Our last analysis concerns the drop-out rate of both synchronous and asynchronous evaluation participants. Fig. 5.7 shows this statistic. The percentages are always relative to the base number of participants ( $n = 15$  and  $n = 72$ ). As it can be seen, here we have the complete participants of both evaluation groups of the evaluated course. When subtracting these drop-outs from the base number of participants, it shrinks to the number of the prior three analytic measures ( $n = 13$  and  $n = 49$ ).

Finally, Fig. 5.8 shows the result of the questionnaire. We aggregate both synchronous and asynchronous participants together and also take into account

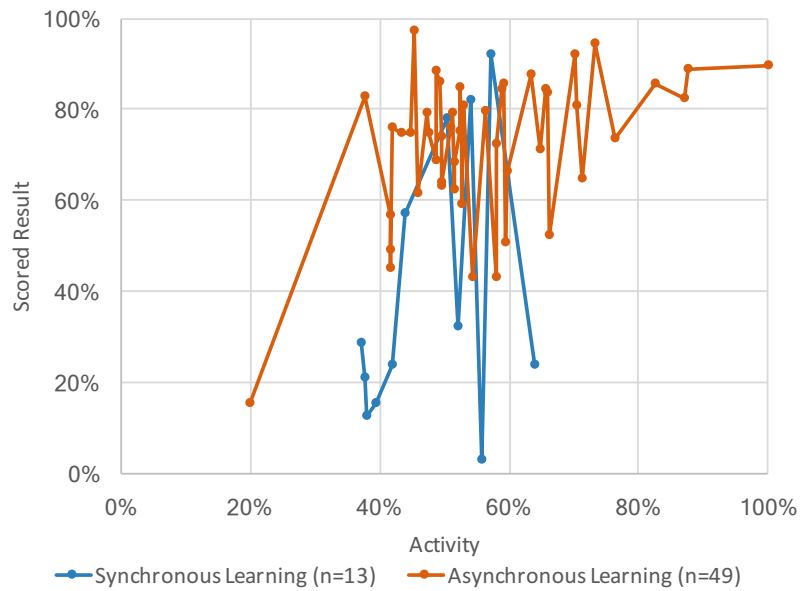


Figure 5.5: V3C evaluation: Activities compared to scored results.

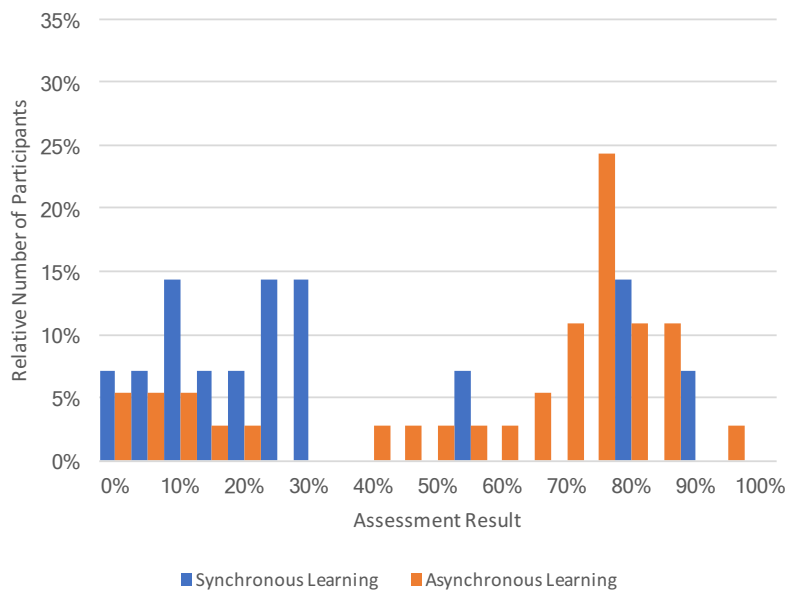


Figure 5.6: V3C evaluation: Assessment results without drop-outs.

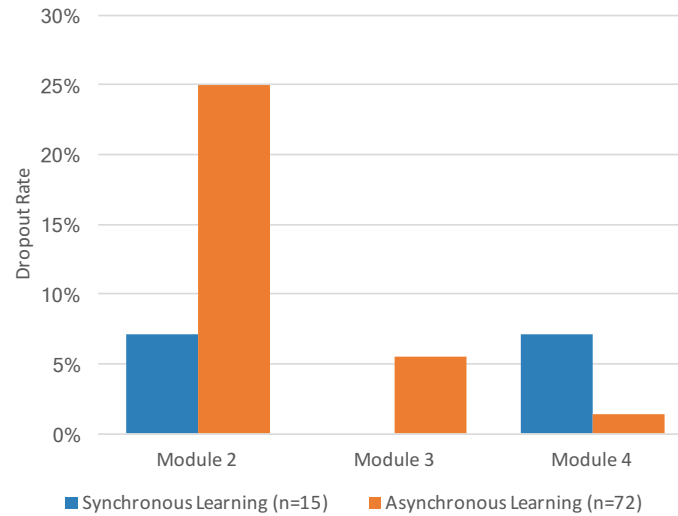


Figure 5.7: V3C evaluation: Drop-out rate.

learners from the course *Social Entrepreneurship*, which we have not taken into account for the prior analysis presented in this section.

### Interpretation

Our evaluation aims at answering two research questions. RQ1 regarded the effectiveness of our technical monitoring solution as well as its capability to make predictions regarding the learning outcome. For the first part, we can state that we were able to monitor all data that was produced by the learners in the evaluation and our platform was capable of handling both synchronous data streams of multiple people collaboratively using the same learning room, as well as that it was robust against a long-term evaluation of two month for the asynchronous evaluation phase. This is also backed by Fig. 5.4, which shows a constant activity flow with the peaks clearly representing assessment deadline phases. For the second part of RQ1, we have to be a little more cautious with our interpretation. For one, Fig. 5.5 shows a rather clear correlation between activity in a learning room and scored assessment results for participants of the asynchronous learning cluster. The Pearson correlation also confirms this with its value close to 0.5, which is considered as rather “correlation confirming” by most literature. For the asynchronous learning phase though, the relation is not that clear. The Pearson correlation again shows there is a slight correlation between both values, but the



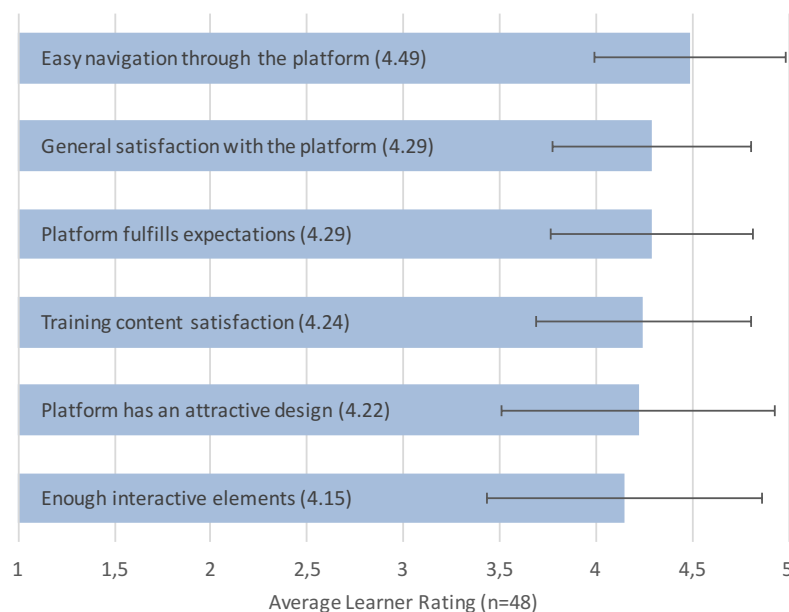


Figure 5.8: V3C evaluation: Questionnaire results.

graph does not really allow for a clear prediction. Reasons for this might be the smaller subset of the asynchronous cluster. With a larger amount of participants here, the results might have been clearer. We also tried to evaluate the results on a per-module basis (instead of the four-module aggregated view), but these presented similar results.

For RQ2, which asked how synchronous and asynchronous learning modes perform if applied in the same platform. What we can clearly state is that there exists a trade-off between time provided to the participants and assessment results. The cluster of asynchronous learners, which had about 75% more time to complete the assessments, scored about 33.38% higher than the cluster of synchronous learners. This can also be seen in Fig. 5.6. On the other hand, it has to be said that reducing the time needed to train people by two-third results also in a very high reduction of costs needed to be spend by employers to provide the training to their employees. Ultimately, it has to be decided by the facilitators of vocational training courses, if this trade-off between the time needed to train people and the expected results is worth taking.

Regarding user acceptance of the platform, our questionnaire results (see also Fig. 5.8) indicate a high satisfaction of the users with the platform itself. Since

all questions received similar high ratings, we will not discuss them in detail, but it has to be said that these ratings might also result from the domain we chose to evaluate our platform. People working or planning to work in the domain of tourism, especially on a service level, might not be used to the support of their training by the means of multimedia in general. So the availability of a platform that provides them with the possibility to perform their training courses online might have been perceived as very appealing, resulting in these high ratings of user acceptance.

### Limiting Factors

Performing synchronous evaluations that span multiple days is a resource consuming task. Especially, when the participants have to be recruited from a particular vocational domain, with full-time job schedules and limited time to spend in training that is not directly financed by their employer. Therefore, we limited the comparison between the two learning modes presented in the previous section to the *Tourism & Hospitality* course, and covered only four out of five modules here. We decided to leave out the duration measure, which represents the time spent for a module, due to some technical uncertainties we had with the results they produced. This is definitely something we need to rethink and re-implemented in possible future work.

Currently, our assessment is limited to the usage of multiple choice quizzes. We are aware of the shortcomings of this approach regarding the capabilities of multiple choice questions to assess learning success as a whole.

During the analysis, we noticed that two learners of the asynchronous phase participated twice in a course, thus we removed both datasets of the participants from the evaluation data. One thing we did find out when perform our analysis was that it would have been interesting to also have a cluster of learners that had the same time to finish the course as the synchronous learners had (two weeks), but without providing them the opportunity to participate in the webinar. Finally, we want to state that our user acceptance questionnaire only aimed at evaluating the platform from a learner's perspective. This results from the fact, that the platform was developed as part of the same project where also the VET trainers that provided the learning content via the platform were partners of. Thus, we had no external VET providers that we could have interviewed to find out about their opinions regarding the model-driven capabilities of the platform, although we deem this an area worth of further research.

### 5.1.5 Summary

We presented an integrated LA approach that allows to compare asynchronous and synchronous learning phases and draw conclusions for certification, interventions and gradual course improvement. It is embedded into a technical platform for the creation, implementation, deployment and performance of virtual training courses. We evaluated our contribution with two vocational training courses, one for social entrepreneurship and one for tourism & hospitality. Based on an English version, we translated both courses in four languages and conducted them with 114 learners from different European countries. Our main focus was the interplay of synchronous and asynchronous learning phases to demonstrate the ability of the platform to blend webinar-style course units with phases of SRL on the platform. Additionally, we had an eye on important factors for both virtual training centers offering courses, as well as companies sending their employees for vocational training, e.g., the drop-out rates. The interpretation of our LA results shows that the combination of webinars and SRL saved additional resources beyond the pure availability of a virtual training center.

## 5.2 An Infrastructure for Community Knowledge Building

In this section, we present our second real-world demonstration and evaluation, which integrates all aspects of the architecture and tool support for decentralized CIS. The outcome, named the “Distributed Noracle”, was evaluated in multiple iterations and integrated most of the infrastructure, scaffolding tools and methodologies introduced in this dissertation. Using *las2peer* as basis for application development, we also used the CAE to create a first scaffold for an initial prototype of the Distributed Noracle. In subsequent iterations, we integrated both the service explorer, the NRT evaluation center and the SBF into the application and the corresponding evaluations.

### 5.2.1 Introduction and Motivation

The vast majority of human learning happens outside of formal settings. Learning activities may be quite informal, as found in incidental learning, SRL and socialization [Schu00]. Some learning may involve more structure or planning, which is generally referred to as non-formal learning [Esha07]. A significant portion of

this learning happens in CoPs. While only few CoPs have the size and influence to get tools tailored to their needs, the long tail [Ande06] of CoPs does not possess the resources, such as central hosting infrastructures or shared budget.

Our use case stems directly from the community described in the introduction of this dissertation (cf. Sec. 1.1). This community of young European youth workers is preparing for participation in a European-funded training course on “creative leadership”. Since the whole CoP neither shares a geographic location, nor central infrastructure or budget, this use case stands exemplary for the needs and challenges of distributed CoPs.

To help establish the boundaries of the participants’ knowledge and identify common ground or potential conflicts, the trainers want to find out which questions the participants have about creative leadership and how those questions relate to one another. Specifically, the trainers implement a form of *Question-Based Dialog* called Noracle [FGMi16] before the training starts, to model and visually represent their common space of ignorance about creative leadership. This special form of IBL starts with a seed question raised by the trainers, which is then answered by the participants by raising follow-up questions. This way, the *Community Ignorance* becomes visible and the trainers gain insight about what the participants are interested in and their views on the subject. As participants create this *Problem Space*, they document the questions they have about creative leadership, their assessments of the questions that others stated and any links they perceive between them. In its analog form, this involves an on-scene session at the start of the training course, where the community has a limited time-frame to establish their community ignorance by writing down questions they have. A digital version of the concept, hosted decentrally by the community itself, could be applied already before the community meets. We state the following two research questions:

- RQ1: Does a digital version affect the community’s knowledge of their ignorance?
- RQ2: Can such a decentralized learning infrastructure be managed by the community?

### 5.2.2 Methodology

We followed the design science methodology throughout the complete process. As it is heavily interwoven with multiple steps described in Chapter 3 and 4, certain overlaps exist, since both the progress of these chapters was influenced by the outcomes of this evaluation, as well as vice versa the Distributed Noracle implemented

newly available features of both infrastructure and tool support developed during the course of this dissertation. Fig. 5.9 describes this process. Our starting point

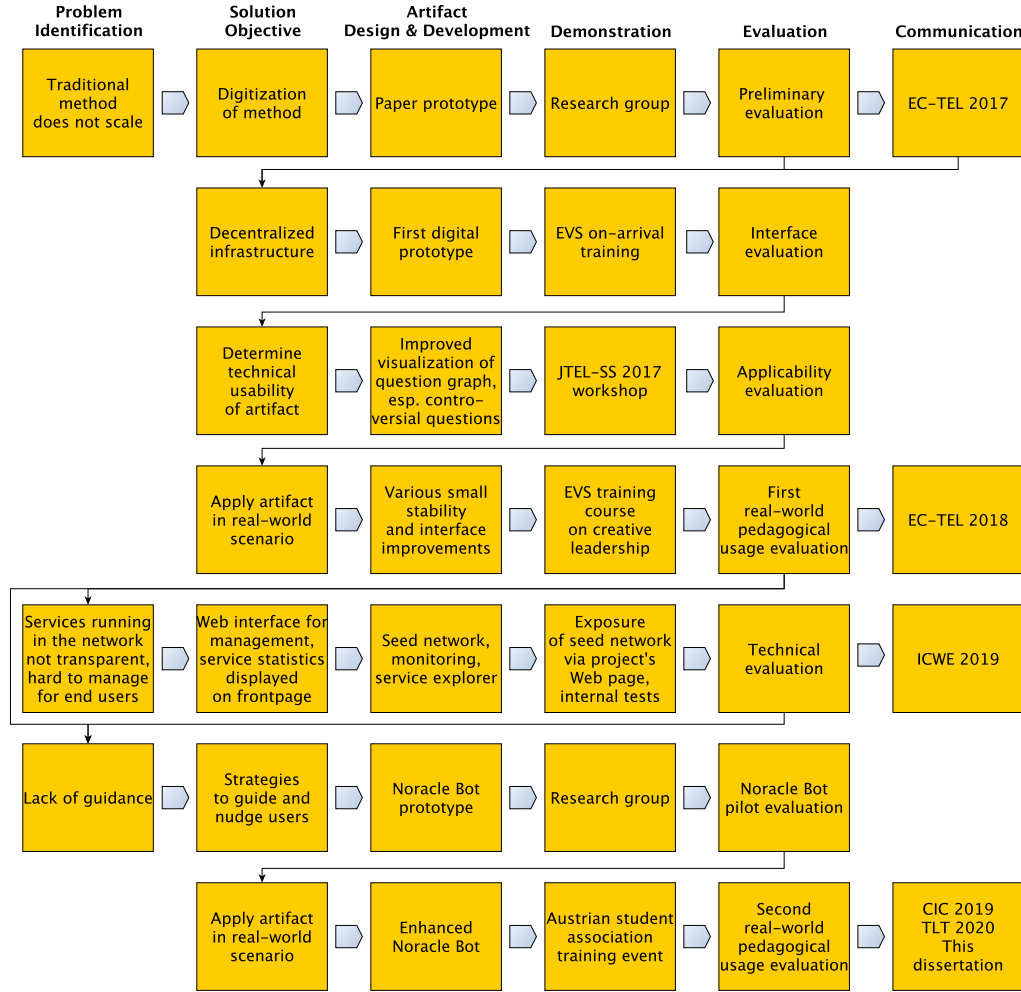


Figure 5.9: Infrastructure for Community Knowledge Building: Design science process.

was the original, analog Noracle method [FGMi16] and its problem of scalability. The preliminary evaluation, based on a paper prototype, led to the requirement of a decentralized version of the application. We communicated these results in a vision paper [LFGK17]. Our next phase was mainly concerned with getting to know how people would interact with our newly developed prototype and the

interface evaluation marks the first evaluation of the digital artifact. We continued with a first evaluation of the decentralized scenario in a workshop setting, which disclosed technical shortcomings we tried to overcome and improve for the next phase, the first real-world pedagogical usage evaluation of our artifact. This rather large evaluation allowed us insights into manifold aspects of both infrastructure and tool usage. The aggregated results of these three iterations were communicated in [LGFK18]. Based on the outcomes of this first real-world evaluation, we found several technical shortcomings of our approach that we addressed in the following iteration. We established the seed network, improved the monitoring facilities and developed the service explorer, which we evaluated in our technical evaluation. These results were communicated in [LJK119]. The lack of guidance, especially with regards to larger question-based dialog spaces was addressed in the following iteration by introducing the *Noracle Bot*, which we describe in our pilot bot evaluation and finally our second real-world pedagogical usage evaluation. Parts of these results build the basis for the use case in [NLK119] and then were covered in detail in [LGF\*20].

### 5.2.3 Digital Question-Based Dialog For Ignorance Modeling

In this section, we describe the functionality of a digital and distributed version of the *Noracle* method. It fulfills our use case and makes it possible to explore and map community ignorance through question-based dialog, asynchronously and without a formal infrastructure.

A space is the main view of the application (shown in Fig. 5.10). Users can create a space and invite others to the space by sharing an invitation link. The user interface provides a list of subscribed spaces such that users can switch between spaces with two clicks. The space view consists of a canvas displaying the questions and their relations as a graph of speech bubbles. It also features a list of users subscribed to the space and a (collapsible) help section. Below the canvas, users can select their current interaction mode. The “Select/Navigate” mode allows users to define the portion of the graph that is displayed. Selected questions and direct neighbors of selected questions are displayed. If a displayed question that is not yet selected has neighbors that would be displayed upon selecting it, they are symbolically indicated as additional speech bubbles behind the question. In the “Drag/Zoom” mode, users can move questions around freely, as well as pan and zoom, to either view parts of the graph in detail or get a birds eye view. The “Add Question” and “Add Relation” mode allows users to add questions or relations by clicking on one question (add a question) or two questions (add a relation). Then,

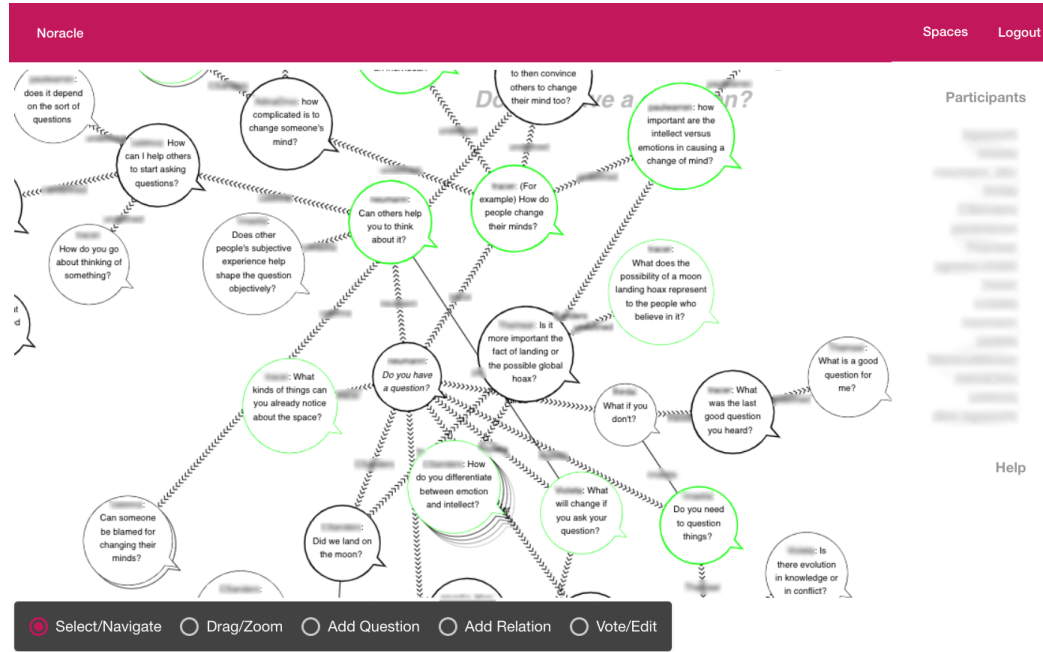


Figure 5.10: Screenshot of the Distributed Noracle application, showing a question-based dialog space used in one of our evaluations.

a dialog window opens that asks the user to enter the text of the question or the type of the relation. For relations, we allow for both *Follow Up* relations (depicted as small arrows indicating the direction), which is the default type of relation that is created between a new question and its parent question, as well as *Link* relations (depicted as straight lines) that display a certain connection of similar questions, although they are not in a direct *Follow Up* relationship. Finally, the “Vote/Edit” mode enables users to either modify their own questions and relations or to assess the value of questions or relations of others. We use a coloring mechanism that displays the entity according to its overall rated usefulness in a specific color, ranging from green to red.

#### 5.2.4 Realizing the Distributed Noracle

In the following, we describe the realization of the Distributed Noracle in more detail. We start with introducing an exemplary usage scenario and continue with a description of the realization of the Distributed Noracle. We close the section by presenting the social bot integration we use in our later evaluations to guide users

through the question-based dialog.

### Exemplary Usage Scenario

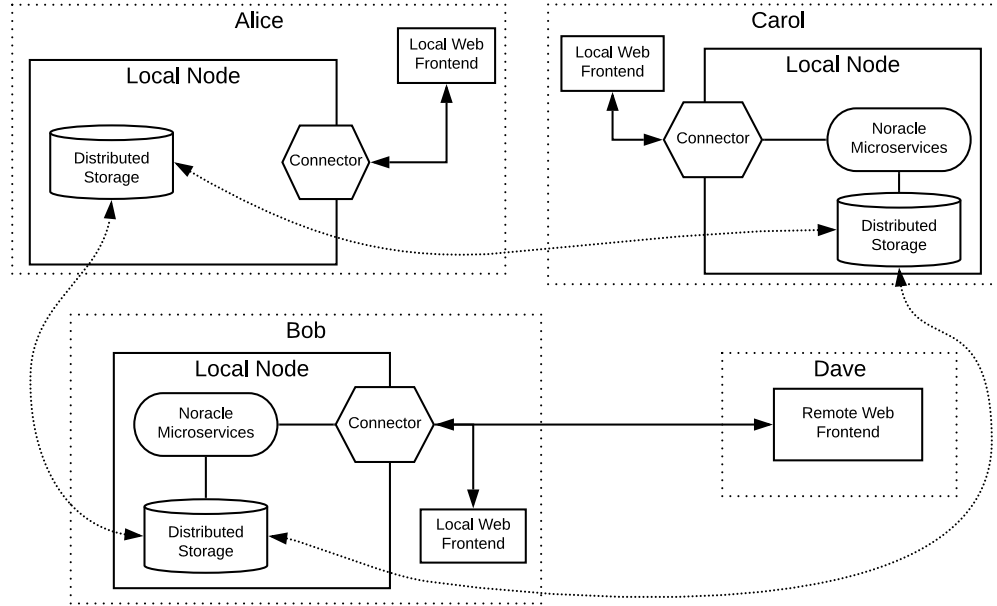


Figure 5.11: Exemplary usage scenario of the Distributed Noracle.

Fig. 5.11 shows an exemplary usage scenario of a Distributed Noracle session, realized on the decentralized CIS infrastructure. While *Bob*'s node features the set of microservices that realize the application, *Alice* has decided to start an empty node without any services running on it. This can have several reasons, including the lack of resources, both in terms of computing power or, especially in mobile settings, energy. *Carol*'s node also contains a set of Noracle microservices, whilst *Dave* has not started a node at all and uses *Bob*'s node to access the remote Web frontend for participating in the collaborative session. As this scenario demonstrates, this approach provides flexible access to the application with several possibilities to join a session. Depending on the currently available resources of a community member, las2peer allows to flexibly start and stop (parts of) applications on a node. This usage scenario does not feature any centralized component, like a master node or a central URL for the Web frontend. Rather, the whole infrastructure is distributed among the community.



### Architecture of the Distributed Noracle

The Distributed Noracle application consists of a set of five microservices that realize different functionalities of the application, and a gateway service to route incoming requests. A *Space Service* handles the creation of spaces and their members. The *Question Service* takes care of creating and updating questions, while the *Relation Service* does the same for relations. The *Vote Service* handles votes for both questions and relations. Finally, the *Agent Metadata Service* is responsible for storing additional metadata (such as the name) for the members of the CoP. Additionally to these five services, the *Noracle Service* serves as the *Gateway Service* to the application that provides a RESTful API to the outside. Being called by the connector, it distributes the requests to the set of microservices we just described. The frontend of our application is based on the Angular framework and it is part of the node, served from the distributed storage. Therefore, we developed a *File Service* that provides a RESTful interface for storing and serving Web frontends directly from the network, removing the need for an additional Web server. Authentication is done using the OIDC standard.

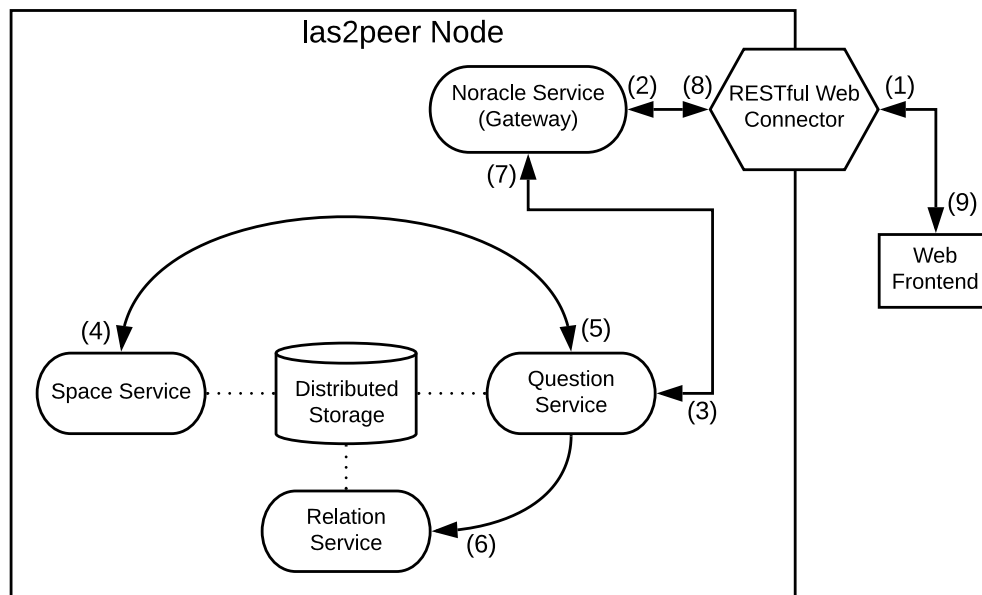


Figure 5.12: A question creation process in the Distributed Noracle.

To give a concrete example of inter-microservice communication, consider

an incoming request for creating a question (see also Fig. 5.12). This RESTful request is transferred from the *RESTful Web Connector* (1) to the *Noracle Service* (2), which sends a request to the *Question Service* (3). This service in turn invokes the corresponding *Space Service* (4) for further details, for example if the user is allowed to create a question in this particular space. Upon receiving the answer from the *Space Service* (5), the *Question Service* creates a new *Question* object in the distributed storage and calls the *Relation Service* (6) for creating the corresponding relation between the newly created question and its parent. Finally, the *Question Service* answers to the *Noracle Service* (7), which sends the reply to the *RESTful Web Connector* (8), who forwards the HTTP Response to the *Web Frontend* (9), whether the question has been successfully created.

This particular scenario is not necessarily limited to a single node, the microservices can be situated anywhere in the network and it is also neither needed nor desired that a particular microservice knows which instance of the called microservice did handle the request. In the exemplary usage scenario depicted in Fig. 5.11, if Alice's node receives such a request, it is distributed throughout the network, because Alice's node does not host any of the application's microservices. Depending on their current load, the request would be processed by the node of either Carol or Bob, and their *Noracle Service* would possibly distribute the just described sub-request again to microservices on other nodes. The flexible scalability of the infrastructure also allows several instances of the same microservice residing at a node, spawning automatically according to the current need. The infrastructure is designed for failure in such a way, that non-responding microservices are automatically shut-down and replaced by new instances.

To provide CoP members with the software needed to start their own node, we created a *Node Package*. It is a small folder that contains an empty node preconfigured to connect to a network via a (configurable) *Seed Node*. It then replicates the microservices of the application via the P2P network and starts them locally. For requirement analysis and feedback, we used the Requirements Bazaar to also include end users in improving the development of the application and underlying framework itself. The complete application is released as OSS<sup>1</sup>.

### Integration of Social Bots

As our evaluations grew larger, also did the resulting question-based dialog spaces. We identified the need for more assistance for users of the tool to navigate their

---

<sup>1</sup><https://distributed-noracle.github.io>

way through the spaces. Thus, the most recent addition to the framework and also the Distributed Noracle application is the integration of a social bot that is capable of sending messages via a chat interface to users, informing them of recent changes to the graph, and possibly interesting areas worth exploring. Here we make use of the concept of nudging [KoHe15], by pointing users to areas in the graph relevant to them, encouraging them to produce content and also to provide relevant information to facilitate reflection.

Although the SBF, following the OSS approach of all software artifacts developed in this dissertation, uses Rocket.Chat as primary conversational interface, the Noracle Bot uses Slack<sup>2</sup>, since our evaluation participants were already familiar with it because of its widely spread use in professional communities. The messages are sent daily and provide information about the community's activity in the Distributed Noracle within the last 24 hours. All questions mentioned in the messages are provided as links directly to the corresponding Distributed Noracle space, with only the linked question initially selected, such that the user starts exploring the graph from this question when clicking on a link in the bot message.

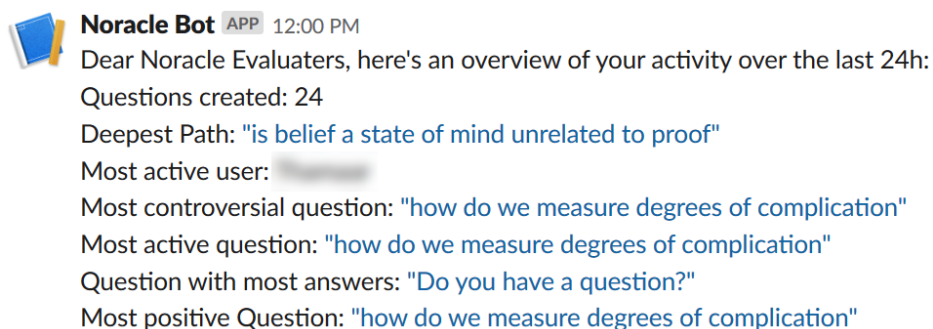


Figure 5.13: An exemplary general bot message of the Distributed Noracle, as it was sent to the evaluators during our pilot bot evaluation.

Fig. 5.13 is an example of the general statistics that the bot sends to a public channel, to be seen by all participants. It starts with the number of questions created, followed by the question with the deepest path, the question that is most distant from the seed question of the space. It is followed by the most active user. The activity includes the creation of follow-up questions, relations and rating questions. The next link directs to the most controversial question, which is the question with the most votes in both directions (helpful/not helpful). Similar to

---

<sup>2</sup><https://slack.com>

the most active user, the message also provides the most active question which caused the most follow-ups relations and votes. Finally, the question that caused the most follow-up questions and the question with the most positive feedback are presented to the community.

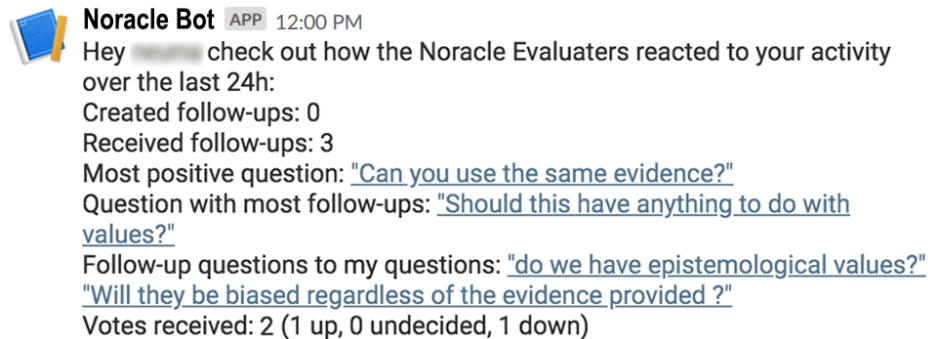


Figure 5.14: A personal bot message of the Distributed Noracle send to an evaluator during our pilot bot evaluation.

Fig. 5.14 shows a personalized message, send to a specific user as a private message. The message gives information about the number of follow-up questions the participant has created and how many follow-up questions the participant's questions have received. In addition, the question that got the most positive feedback, the question that raised most follow-up questions and each individual follow-up question of the user's questions are displayed. Finally, the number of votes received is provided.

### 5.2.5 Evaluation

We evaluated our application in multiple iterations, with different types of learning communities. Following the design science methodology described in Sec. 5.2.2, each evaluation had a certain focus that lead to a gradual improvement of our approach and implementation. In the following, we present these evaluations and finish this section with a discussion of the outcomes.

#### Preliminary Evaluation

In the preliminary evaluation, a Web science research group used a paper mock-up of the Distributed Noracle for questioning current priorities in their research field. The purpose of this evaluation was to determine whether the method could

be transferred to a digital space and which features would be required. This community was appropriate because of the shared interest in a topic, diverse levels of experience, and a loose collaborative structure.

**Participants and Procedure** 8 members of the community took part in the trial. Half of the participants were more experienced members of the team, as determined by whether or not they were supervising PhD students. The other half were PhD candidates or post-doctoral researchers. To represent a shared digital space, the participants worked asynchronously on a large poster in the lab. A general reflection question was posed as the seed question in the Distributed Noracle mock-up: “What is the most relevant, open question for social semantics?” Each participant received a differently colored marker to represent her contributions to the poster. As participants added questions, they were also asked to circle questions they supported and draw links between questions to show their relationship. Participants also starred those contributions they thought were most helpful. The evaluation lasted for three days.

**Analysis and Outcomes** After concluding the exercise, the participants took part in a short group discussion regarding the insights they could draw from looking at the question graph. The main themes of the discussion were:

1. The tool could help to structure dialog more efficiently.
2. It encourages to consider broader or new perspectives.
3. Participants need assistance in interpreting the graph.

The participants also expressed thoughts about the overall value of the proposed artifact. They emphasized the additional possibilities a digital version would provide in terms of longer running efforts to structure their thoughts as a group. The need to transfer the process of question-based dialog to a digital space to increase its value was established through this evaluation.

### **Interface Evaluation**

The first evaluation of the digital tool was conducted with participants on an “on arrival” training for participation in the EVS program. The participants used the Distributed Noracle to consider the future of European youth work in the context of a project planning session. This community was appropriate because of the

ill-defined nature of the topics that participants were exploring and their lack of having a central infrastructure.

**Participants and Procedure** 7 participants between the age of 20-25 from different European and Erasmus+ partner countries took part in the study. The participants had similar levels of experience in the area of youth work (1-2 years). In this evaluation, the participants worked synchronously. All participants used a given link to access a single-node deployment of the Distributed Noracle. After a project planning session in their face-to-face seminar, the participants joined the space and continued their reflections online. They had a set period of time to explore the application with the general reflection question posed to them “What is the future of European Youth Work?”. As participants added questions, they were also asked to assess questions they found helpful and create links between different questions to show their relationship. The exercise lasted for approximately 30 minutes, followed by semi-structured interviews with the participants regarding usability and value of the digital tool.

**Analysis and Outcomes** Since this was the first evaluation of the digital version we focused on the aspects that inherently differed from the face-to-face version. In particular we looked at analytic features designed to help the individual to get a sense of a question’s *importance, quality and validity* for the group. Examples for this are the marking of questions where conflicts are present in red, or darkening the circle that surrounds the topic as more and more contributors agree that the question is relevant. All participants agreed that they understood the semantics of these analytic features. Some participants suggested improvements regarding the visual representation. The most frequently mentioned suggestions for improvements concerned the layout and animations of the graph itself. Some participants considered the automatic force-directed layouting system slightly disorientating at times. We attributed this to the prototypical nature of the design artifact in this first evaluation and improved the overall look and feel in further iterations.

### **Applicability Evaluation**

The applicability evaluation was conducted with workshop participants of the Joint European Summer School on Technology Enhanced Learning (JTELSS). The purpose of this evaluation was to test the technical features of the tool, in particular the decentralized architecture and its applicability within a group of

people with diverse technical and non-technical backgrounds. The community was considered appropriate for a this kind of evaluation because of their experience with educational software.

**Participants and Procedure** Approximately 20 people attended the workshop. First, the participants were given a short introduction to the method of question-based dialog and to the application. As part of this introduction, participants were instructed on how to start their own node and join the network. Participants used their own devices to launch their nodes. We provided a local seed node that participants could connect to. The participants were then given about 20 minutes of time to explore the tool. We provided a general starting question in a sample space. Participants were also asked to assess questions they found helpful and create links between different questions to show their relationship. In addition, they were invited to create their own space and invite other participants to join. In the end, we asked participants to fill an online questionnaire.

**Analysis and Outcomes** We received 7 questionnaire responses. Participants shared the same conceptual understanding about the possible usage context of the tool: 5 responses were similarly themed around conceptual mapping, development of a common understanding and knowledge expansion. The visual representation played a significant role: 4 participants expressed the importance of question-color and -size. Regarding usage monitoring, we were able to capture data from 12 workshop participants, of which 8 participants started their own node that connected to our seed node. This network of nodes also provided access to the application for the other 4 participants, who were unable to launch an own node on their device. They used the Web frontend provided by another participant's node (cf. usage scenario *Dave* in Fig. 5.11). This demonstrated the capability of our approach to overcome technical hurdles, such as firewall restrictions or device security policies. The data we received from this evaluation was afterwards used to improve the application, leading to a more stable version used in our first pedagogical usage evaluation.

### First Real-World Pedagogical Usage Evaluation

The first real-world pedagogical usage evaluation was conducted with the community described in the initial use case (cf. Sec. 5.2.1). Participants of an European training course on creative leadership were invited to take part in an experiment

using the Distributed Noracle to help prepare for the course and explore their existing knowledge gaps about the topic. For the organizational team hosting the training course, an orientation activity of this kind is conducted typically at the beginning of a course. Depending on the methods used and the complexity of the course, this can take several hours to achieve with participants. The incentive for the host organization to take part was in improving the on-boarding process for participants and gaining an initial understanding of their ideas. The purpose of this evaluation was to test the application in a real asynchronous and distributed setting, adding monitoring data to the qualitative verbal and written data.

**Participants and Procedure** 34 participants took part in the evaluation. The participant group was diverse, with different nationalities, levels of experience and knowledge about the subject of the training course, *Creative Leadership*. One week before the training course, participants were notified via email that an “experiment” would be taking place, using a beta version of an application to help prepare for the training. They were informed that their participation in the experiment was completely voluntary, but that the training team and researchers felt the tool could be helpful in establishing what this particular group of participants found most confusing or difficult about the concept of creative leadership. The participants received information on how to join the Distributed Noracle and were invited to contribute their own questions to a specific reflection question (“What is creative leadership?”).

Since the participants were locally distributed with prior contact only via email, we created an artificial distributed setting by creating a network of nodes at a university. We provided a URL to the participants that automatically distributed them to their specific node. This created a scenario where each participant had her own node, without the actual need for a technical setup procedure that would have been unfeasible for this particular evaluation, especially regarding the evaluation of the results. After the first 48 hours, participants were asked via email to review the questions that other participants had posted so far and evaluate how important or useful they are to the overall discussion. Participants were also encouraged to make links between questions and add relevant follow-up questions to the questions of other participants. Once the participants arrived at the training course, the entire trainer team and the trainees participated in an analysis of the question graph and an evaluation of the tool’s features. The evaluation included three items: *What insights can you draw from the graph? What features or functions might improve the value of this tool for you? In which situations could you imagine to use it?* Each



individual had five minutes to review the graph and to take some notes. Then, the facilitator gathered the insights in a plenary session, during which the participants' statements were also clustered according to their shared theme.

**Analysis and Outcomes** With regard to the insights that could be drawn from the graph, the participants found it quite easy to see what was most important to the group, such as focusing on the development of *creative skills*. When asked how they evidence this with the graph, the participants noted that many questions related to this topic in some way. The graph also showed a considerable agreement about the importance of questions related to this topic (as indicated by the green color).

Participants stated, that the graph helped them to realize they had taken a *very individualistic perspective on creativity and leadership*, with very few questions having to do with social aspects of creative development. Considering that a large part of the training course was founded on shared leadership and joint creativity, this was important to the training team to have highlighted in the graph. The way that questions were formulated allowed the participants to differentiate between questions related to defining creativity ("what" questions) and questions related to the process of developing or improving creativity ("how" questions).

The trainees agreed that the tool helped establish the interests of a group in advance. This is useful in a variety of settings, in particular educational settings that are blended or fully online. They also felt that participating in the experiment was a valuable use of their time. Using the tool in this way saved the training team an estimated three hours of time with the participants on the training course, allowing them to more quickly engage with the subject.

The training team remarked, that instructions were important in helping the participants to know how to use the application. Especially with new users, they recommend facilitation to maintain the quality of the space by demonstrating question-asking and some of the application's additional features. Features that participants felt were important to develop had to do with analytics to help uncover other types of insights or consequences. For example, only one trainee had noticed that similar questions were repeated several times in the graph. When the training team highlighted this point to the group, they agreed that this was valuable information that they missed as the graph became larger and less visually manageable. In addition, a third of the participants said that they would find it helpful if there was a way of knowing exactly how many people or a percentage of people found a question useful. Participants felt this information could help them

identify questions they might find interesting or important. All of the participants and the trainer team felt that the tool would be improved by having a way of visualizing what insights or consequences could be drawn, for quick and easy reference.

Beginning with this evaluation, we also began live monitoring the complete network for user activities. We started the monitoring the day we sent out the invitation mail, while we asked the participants to start their 48h collaboration phase on the beginning of day three. With the help of the monitoring data, we recorded high activity between day three and five, while it declined afterwards. Still, the number of recorded activity before and after this “official” trial phase shows the *intrinsic motivation* participants had to (re-)visit the problem space, an important factor for learning activities in SRL scenarios. Another interesting observation we made during analyzing the monitoring data was, that with an average question depth of 1.9, a question was on average about two questions away from the seed question. We perceive this as another indicator of the usefulness of the graph-based visualization, since most questions did not connect directly to the seed question, but to follow-up questions, demonstrating the evolving awareness of the community ignorance, represented by the growth of the graph.

After this evaluation, we implemented the service registry with its service explorer frontend (see also Sec. 3.4.3) to allow non-technical community members easier handling of services and applications via a node-frontend, while also verifying services running in the network and detecting the upload of malicious services to the network. We then continued with a technical evaluation to measure capabilities of our framework and application in terms of larger spaces and high usage simulations.

### Technical Evaluation

Apart from our user evaluations, we also evaluated the Distributed Noracle application from a technical point of view. This work was mainly done in between the first pedagogical usage evaluation and the integration and evaluation of the Noracle Bot, but we continued to monitor usage activity thenceforward.

**Procedure** From the first pedagogical usage evaluation on, we maintained a network of ten always-online “seed” nodes. This served two purposes. On the one hand, these nodes acted as an entry point to the Distributed Noracle application directly by providing multiple Web connector endpoints that are displayed on our project’s Web page. On the other hand, they allowed to connect one’s own node

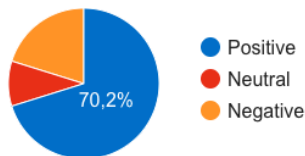
to the network to participate. Additionally to these nodes, we installed a MobSOS monitoring node that collected (anonymized) usage statistics of the Distributed Noracle. The pedagogical implications of this monitoring data already were partly covered in Sec. 5.2.5. But from there we went one step further and added pre-processed monitoring data to the landing page of the application’s frontend, using the NRT evaluation center. This way, each user that either started her own local frontend, or used one of the provided ones, was able to see them.

Additionally to this, we wanted to test the technical capabilities of our application and framework in terms of data processing and provision. Therefore, we took a snapshot of the data of a Reddit megathread<sup>3</sup> and processed it in a Distributed Noracle space, measuring the time it took. It consisted of 163 users, posting 308 questions and answers. We interpreted each response, regardless of its type (question or answer), as a follow up question to its parent.

### Usage Statistics

Number of Questions: 626  
 Number of Spaces: 38  
 Average Questions per Space: 16.47  
 Average Longest Thread per Space: 3.23

#### Question Votes



#### Created Questions

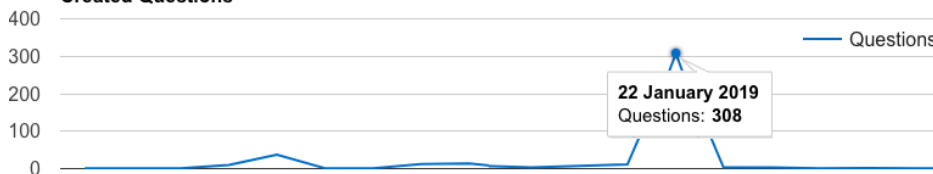


Figure 5.15: Usage statistics of the one-year decentralized infrastructure deployment, continuously providing the Distributed Noracle application since our first real-world pedagogical usage evaluation.

<sup>3</sup>[https://www.reddit.com/r/NoStupidQuestions/comments/addb81/us\\_government\\_shutdown\\_megathread/](https://www.reddit.com/r/NoStupidQuestions/comments/addb81/us_government_shutdown_megathread/) (Snapshot taken on 22.01.2019)

**Analysis and Outcomes** Fig. 5.15 shows a screenshot of the usage statistics collected in a roughly one-year period since the start of the first pedagogical usage evaluation. Clearly visible is the peak of created questions on the day we uploaded the Reddit data to the network, making up for about half of the 626 created questions in this time period. Interesting observations are that the majority of people rather rate questions as helpful (positive) than neutral or not helpful (negative). The average (median) number of questions per space is quite low with only 16.5 questions stated per space. We interpreted this as a clue that guidance and nudging support, as we introduce later with the addition of the Noracle Bot, are crucially needed to handle larger spaces.

Regarding the Reddit megathread, the time it took to create all 163 users was about 3 minutes, whilst the creation of the 308 questions took about 100 minutes. It has to be noted, that the latter time is also the result of not being able to parallelize the process, since a question could only be uploaded once its parent question existed, so that it could be linked to it. Nevertheless, in a real use case of such a big thread, multiple questions could be created in parallel, since many questions have the same parent. This evaluation showed that the framework is technically capable of processing large question-based dialog spaces. However, Fig. 5.2.5 shows a (partial) screenshot of the resulting graph. As one can see, the space is very crowded. Apart from the more-or-less obvious seed question in the center of the space, it would be very hard for users to find relevant areas in the graph. Also, the visualization tended to re-render pretty often due to the large amount of questions to be displayed on the screen. Especially in mobile usage scenarios, a large space like this would become impractical to use.

On the basis of the now available technical evaluation data and the newly implemented monitoring capabilities, we thought about ways of providing relevant analytical information about the space to the community. The approach that we felt made most sense to focus on, given the pedagogical and technological underpinnings of the Distributed Noracle, as well as the improvements suggested by participants, was the introduction of social learning bots. The diversity of participant needs suggested that this new facilitating tool should guide the users through the problem space, tailoring themselves to the user by analyzing the previously monitored usage data. Bots offered a seamless choice for providing this service on top of the existing application and framework.

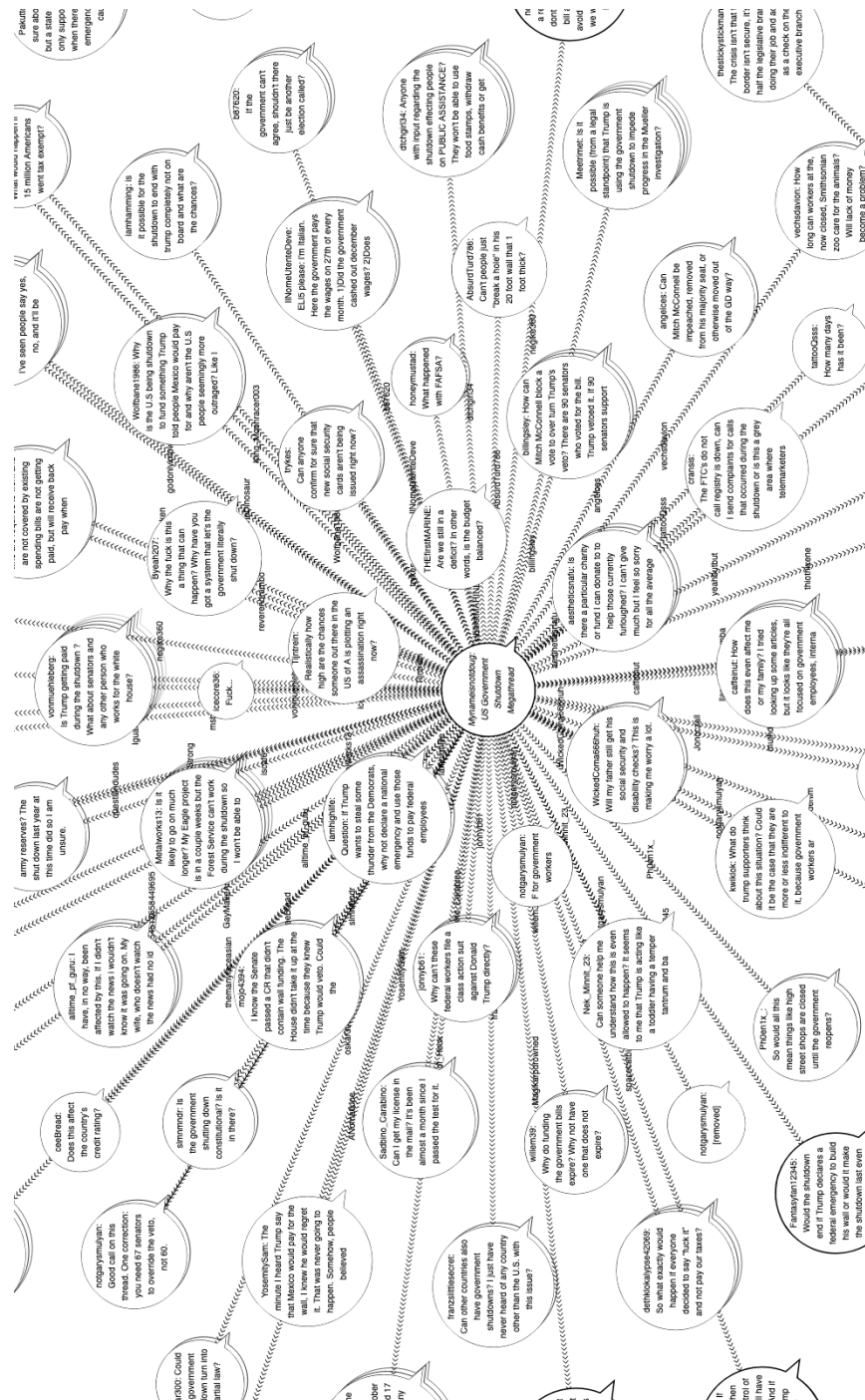


Figure 5.16: Visualization of a Reddit megathread as a Distributed Noracle space.

### Noracle Bot Requirement Analysis

To gather requirements on what type of information the Noracle Bot should report, we conducted a small study with five educational experts from the domain of our first pedagogical usage evaluation. While we do not see this study being an “own” iteration of our design science process, it provided us with the requirements that influenced the development of the pilot bot iteration described in the next section.

**Procedure** In this study, we told the participants that we intended to create an analytical tool to help users engage with the Distributed Noracle application. We briefly explained the concept of a social bot, so that participants would understand how to frame their requirements. We then asked them to engage with the tool as part of a regular reflection activity and to observe the space, the users, and the content they create. Following the activity with the Distributed Noracle, we delivered a guided meditation, asking participants to consider what could happen if the number of questions in the graph were to increase ten-fold, or if more users were to participate. Finally, we asked them what they believed might happen if users were engaged in multiple spaces with different types of seed questions at the same time.

**Analysis and Outcomes** Participants mentioned capabilities that the bot should have including informing you if someone interacted with one of your questions or other contributions in the space. In addition, participants were interested in seeing high quality information that helped them understand and navigate the space. For example, participants were interested in which questions provoked the most activity, agreement or disagreement in the space, as well as which question threads appeared most dense. Finally, the bot should provide both general and personalized reports related to different participants’ activity in the space.

### Noracle Bot Pilot Evaluation

On the basis of this requirements analysis, we created the *Noracle Bot*, a social bot for the Distributed Noracle, and conducted a pilot evaluation to test both the capabilities of this first bot prototype, as well as our evaluation procedure for the upcoming second real-world pedagogical usage evaluation. Following recommendations from the first pedagogical usage evaluation, we facilitated a Distributed Noracle space for this pilot evaluation and also included participants who were familiar with the face-to-face or digital versions of the Distributed

Noracle. In addition to facilitation by these researchers, these influencers' role was to help shape the activities of the pilot "community". Participants were invited to use the space as they might if they were actually working together on a topic, and to observe the functionality of the Noracle Bot as the question-graph expanded.

**Participants and Procedure** 9 participants from the researchers' networks of past participants and colleagues took part in the evaluation, with one participant being in a location with an internet firewall that rejected the application and thus not being able to participate. The participant group was diverse, with different nationalities, levels of experience, domains of activity, age and knowledge about the method of question-based dialog used in Distributed Noracle.

We provided a dummy seed question of "Do you have a question?" to allow participants to choose which themes they wanted to bring up in their questioning. After receiving instructions on how to participate in the Distributed Noracle space, the evaluation began. We also asked participants to use the tool for five days during the work week and to make at least three contributions per day.

At the end of the week, we prepared an anonymous online evaluation for participants and invited more in-depth feedback both in-person and via e-mail. The questions provided to participants in the evaluation were intended to refer back to the basic requirements that previous participants mentioned: to provide more information about one's own activity in the space and to identify points of interest in the graph, delivered in a way that users find manageable.

**Analysis and Outcomes** Three days into the evaluation, the question graph grew so large that participation became very difficult. However, the evaluators attempted to continue until we stopped the evaluation exercise one day early. Six out of the eight participants who were able to participate in the evaluation had positive impressions of the Noracle Bot from the evaluation, rating the quality of information provided as "high" or "very high". In addition, the majority of participants believed that the bot was most likely necessary for navigating the space effectively. With regard to the bot's ability to provide information about one's own activity in the space, seven participants described it as "valuable" (one participant offering a modifier of "highly valuable"). With regard to helpfulness in identifying points of interest, the bot performed slightly worse, with six participants describing the bot's performance as "helpful" and both one modifier of "very helpful" and "neutral". Concerning the delivery of messages, the diversity of

participant answers indicated that in the future, the timing of the bot's messages and the exact information it provides should be customizable. With the open questions that we provided in the evaluation to gather more qualitative data, we learned that the participants did evaluate the Noracle Bot as being mostly successful at the capabilities it was designed to do and that participants found those things useful in similar ways to those of the participants in the requirements evaluation. One particular interesting improvement mentioned was to have the "flexibility in when to call the stats and a deeper look at how many branches you created or how many child relations came from one of your questions".

### **Second Real-World Pedagogical Usage Evaluation**

Our second real-world pedagogical usage evaluation took place in the context of an Austrian student association training event for young people that lasted several days. The trainers organizing the event were in a generational change, with some of the more experienced trainers on the edge of retirement and some novice trainers participating for the first time. Also the handover of the event management, which took place after this years event, was announced well in advance. With this background, many discussions were held about the future direction of the event. Noracle was chosen as a method to support these discussions and open them up to all staff members. We defined *three main areas of discussion* and formulated open ended questions to initiate *three predefined Distributed Noracle spaces*. The evaluation participants had the goal to create a comprehensive, collective understanding of what their burning questions for the future of the event were.

**Participants and Procedure** Evaluation participants were ten trainers from this event. The participants were male, aged between 20 and 30 years with different professions. The event consisted of lectures, discussions, practical activities, and evening events. Every trainer had a different schedule and different tasks during the day, such that face-to-face group reflection was only possible after 10pm. The event started on a Friday at 4pm. Scheduled event activities, debriefing and preparations for Saturday kept the trainers busy till midnight. After that, the Distributed Noracle was introduced as a tool to facilitate collaboration and group thinking, despite the different schedules. Technical support and guidance was provided to ensure that the system was working properly for all participants. They were instructed to use the Distributed Noracle for the coming days of the event, and that they would receive social bot messages regarding the activity of the question-based dialog via Slack. The social bot was configured to send out messages at 10pm such



that participants would also be able to discuss the provided information. The evaluation ended on the next Wednesday's morning and participants were asked to fill a questionnaire.

**Analysis and Outcomes** The participants started creating questions on Friday night. The last question was asked on Tuesday afternoon. Within this time frame 54 questions, linked with 58 relations, were asked and 22 votes were cast. Of those 54 questions, 34 were created between 8am and 10am, 10 between 4pm and 6pm. Between 10am and noon, as well as between noon and 2pm, 3 questions were created each. Finally, 4 questions were created between midnight and 2am, all on the first night after the tool was introduced. This distribution of activity aligns with the expectations we had from the different schedules of the trainers. As the questions were asked over a period of about 85 hours, on average an average of 15 new questions were asked per day. Most activity was recorded on the second full day of activity (Sunday). The ratio between new relations and new questions rose from 0.75 new relations per new question on Saturday to 1.19 on Sunday and peaked at 1.38 on Monday before falling to 1 on Tuesday.

To cover the difference in perception between single-space and multi-space usage, we extended the questionnaire from the pilot evaluation, such that we could compare the perceived differences in terms of sense making of what is happening, understanding ones own activity and pointing to questions of potential interest. Eight questionnaires were handed in. With respect to the multi-space evaluation, participants perceived most value of "the Noracle Bot in pointing you to questions of potential interest" across spaces, as compared to a single space scenario. Two respondents perceived the Noracle Bot very valuable in this regard across multiple spaces, but only somewhat valuable for a single space. Another respondent perceived it extremely valuable across multiple spaces and very valuable for a single space. The other five respondents found it equally valuable. This observed difference in perception on single and multi-space scenarios is not significant, but well aligned with the intuition that the value of a bot raises with growing complexity.

From the open ended questions we extracted the following reoccurring themes from the responses (occurrences in brackets):

- Question-based dialog as method triggers thinking in new structures (6). It was noted however, that it may not suit every problem (1).
- Bot messages were informative (6) about structure (4), where questions belong and connections should be modeled (2).

- Information provided by the bot needs to be (more) precise/specific (3) and directly related to ones own questions or actions (2).

Responses to whether or not the participant would recommend using the Distributed Noracle to a friend or colleague, resulted in two detractors (those who would not recommend), one passives (who would not agree to recommending or not recommending) and five promoters (who would recommend).

## Discussion

Improvements proposed by users mostly dealt with the interface and analytic features, such as additional ways of visualizing other aspects of the dialog by making nodes larger or smaller, allowing for certain questions to be marked as “resolved” and additional ways of linking questions. Most of the users in all evaluations said that such a tool can be useful in the planning stages of a project and at the beginning of any complex task or assignment to gain orientation. In addition, participants saw affordances for structuring group- and teamwork in schools. Improvements suggested by the evaluators also included providing more information about the tool and how to use it, to make the graph searchable by keyword, and improving the interface.

From the technical point of view, the evaluations showed potential weak points of our application, such as the stability and ease of starting a node. While we were able to solve many technical challenges and improve the system during and after each of the evaluations, we are still working on improving both points. Nevertheless, the two real-world pedagogical usage evaluations proved that our prototype is applicable in real-world usage scenarios.

The trainer team of the first real-world pedagogical usage evaluation stated they were able to save considerable time in gathering important information on the trainees’ expectations and knowledge. In a typical training scenario, a half day would have been spent on these types of abstract questions about the program. In this case, it only took 45 minutes of analyzing the resulting question-graph to achieve an even better result. In addition, starting the process in advance seemed to have the effect that the group took the exercise more seriously, which lead to these better results. Possible reasons for this mentioned by the trainers were that when the method is used in face-to-face settings, the participants are naturally distracted by the person they have in front of them. The tendency to move towards providing answers or advice makes it more difficult to keep them on task. Working asynchronously with the participants appeared to have resolved this as it was not necessary to always repeat that the participants should only ask questions.

Our second real-world pedagogical usage evaluation did show the helpfulness of the Noracle Bot, especially in a multi-space scenario. This is still a topic to be explored further, since we only scratched the surface of the possibilities social bots bear, and we did not evaluate the (potential) role a social bot plays in the community itself by being a real member of the space and, e.g., creating questions based on previous domain knowledge. These processes bear a lot of potential for further investigation of user guidance and nudging in CoPs.

### 5.2.6 Summary

With the Distributed Noracle, we demonstrated and evaluated the application of a decentralized CIS for knowledge building. The infrastructure, based on the technical and methodological contributions introduced in the previous two chapters, can be managed by the community members themselves. The microservices realizing the applications are based on las2peer, and thus are able to self-replicate through the network according to the community's current needs and provide the necessary information. The introduction of the Noracle Bot, as an exemplary utilization of the SBF, connects the decentralized learning environment infrastructure to already established tool support in CoPs. Our real-world pedagogical usage evaluations proved the applicability of our approach in the domain of non-formal learning communities. The evaluations also showed that a digital version of a proven method for IBL affects the community's knowledge of their ignorance, which answers our first research question we posed, and can be build with our presented infrastructure, managed by the community itself, answering the second research question. The Distributed Noracle provides both potential time-saving opportunities, as well as it enables question-based dialog sessions that would otherwise just not be applicable because of spacial differences within the CoP, or the lack of centralized, managed infrastructure. Our approach concentrates on taking into account the specific attributes of CoPs, like temporal and spatial dynamics. By consequently addressing these attributes, we support CoPs in their efforts to share and acquire knowledge. As information remains available throughout the communities' existence and services evolve continuously at the same time, our infrastructure ensures sustainability and adaptability, aptitudes we reckon to be crucial in the development of a more democratic and egalitarian Web.

## 5.3 A Distributed Mentoring Architecture

We finish this chapter with our third and final demonstration, which provides an outlook on future developments of the methodologies and tools developed during the course of this dissertation. Set within a more formal learning scenario, we embed the approaches presented in Chapter 3 and Chapter 4 in a larger, shared infrastructure for digital university mentoring. This still ongoing effort is conducted in the scope of a long-running German research project. The testbed-based structure of the project allows for large evaluation setups with hundreds of participants, of which much of is still ongoing work.

### 5.3.1 Introduction and Motivation

Mentoring is the process of the mentor supporting the mentee, in order to make the learning experience more effective and efficient. Psychological and emotional support are at the heart of the mentoring relationship, underpinned by empathy and trust. Mentoring can provide multiple roles [RiSa12]: counseling, instruction, training, activation, motivation, socio-emotional support, networking and example. There are also other success factors that make mentoring effective, like similar values, demographic proximity, trust and respect. Many of them have been already considered in existing approaches, including affect detection, meta-cognitive support, lifelong mentoring or prediction [KSIg19].

In modern higher education institutes, mentoring has become challenging due to the mass of students and the lack of resources. It has raised the interest in socio-technical support for mentoring processes, which include peer mentoring and technological processes. Intelligent Tutoring Systems (ITS) [BrPe03] have already a rather long tradition in university teaching, and their role as virtual mentors in mentoring processes has lately been recognized [HoWa15]. Topics like peer mentoring, virtual mentors, affective and emotional support [TGDN18], but also minorities [MCH\*19] and modeling [DiBr16] are subject of recent research.

These topics were successfully applied especially in those areas, where domain knowledge can be well formalized with the help of experts. However, motivations, emotions and meta-cognitive competences play a crucial role in education. They can be monitored through big educational data and a wide spectrum of available sensors, bearing the potential to also improve the mentoring process.

In this application of the methodologies and tools developed during this dissertation, we take a look at these various aspects and investigate how they can be technologically supported, in order to satisfy the requirements for Intelligent

Mentoring System (IMS). Our use case for building the infrastructure mainly stemmed from an accumulation of requirements that emerged during the course of a project funded by the German Federal Ministry of Education and Research (BMBF), called tech4comp<sup>4</sup>. In this project, a consortium of multiple universities works together to develop a shared infrastructure for distributed, digital mentoring processes. As each partnering university employs different LMS, as well as a variety of multiple other data sources like sensors, questionnaires and interview/face to face mentoring data, the need for a unified representation of this data for further processing arises. Also, the combined efforts for a shared infrastructure brings up questions with regards to data protection and privacy. Finally, as each partner uses different systems for both generating the LA data used for mentoring, as well as to actually provide the mentoring, the need for a unified interface for mentors and mentees arises. While this project is not the only source of requirements, and there are a few other research projects that now actually use the presented infrastructure, this described usage scenario is the largest for now. In the long run, with this demonstration we seek to answer the following research questions:

- RQ1: How can we design an infrastructure to exchange data between universities in a private and secure way to scale up on the inter-university level?
- RQ2: How can we integrate heterogeneous data sources to facilitate services supporting mentoring processes?

### 5.3.2 Architecture

Fig. 5.17 gives an overview of our current infrastructure. Applications are installed and operated within a kubernetes cluster. Handling of the data is done decentrally by a las2peer network. This network resides partly within the cluster, as well as on premise of the corresponding LMS, to ensure that the complete data transfer is protected by asymmetric encryption. The blockchain-based LA verification service (cf. Sec. 3.6) resides within the cluster. Our infrastructure is connected to the learning toolchain via so-called *Data Proxies* for different LMS, which are located at the institutions of the respective testbed. These form the external part of the las2peer network, with the task to transfer the data from the respective LMS to the cluster. Currently, this is integrated for the LMS Moodle and Opal. After running through the verification system, the incoming data flow from the data proxies is aggregated within the cluster via a monitoring pipeline and streams into a collection

---

<sup>4</sup><https://tech4comp.de/>

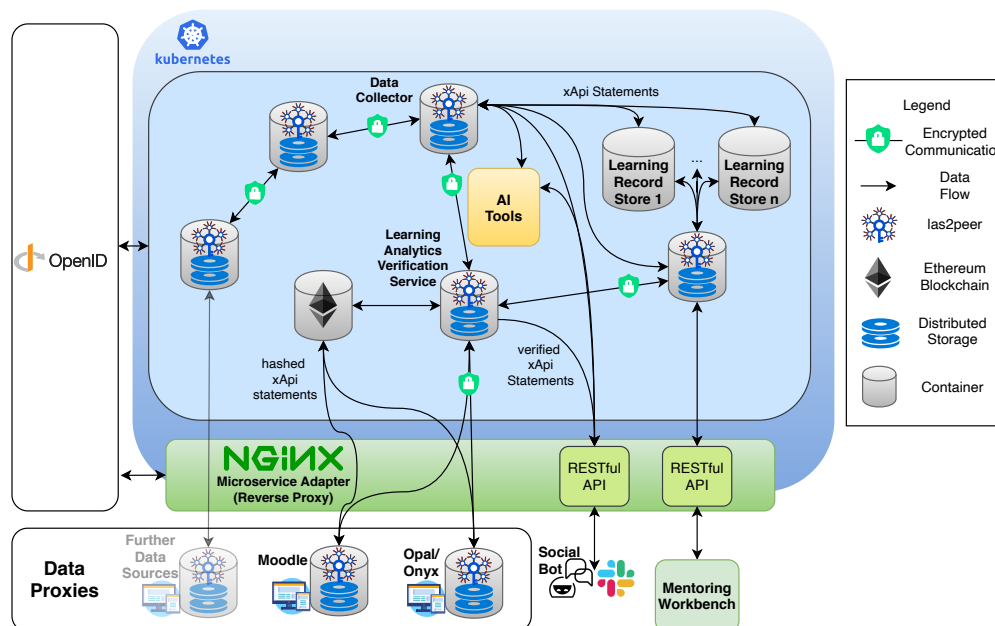


Figure 5.17: Mentoring support infrastructure.

of connected LRS. This data is currently analyzed by basic knowledge services, which provide traditional models of domain, learner and pedagogy, based on both rule-based and machine learning approaches. The service registry (cf. Sec. 3.4) allows for securely archiving these services with different versions, which in turn makes it easier to find and access them. In the future, smart assistance services will use this data to implement a spectrum of supporting functionalities, including personalized recommendations, categorizations, predictions and reflections.

As interface for both mentors and mentees, we used the SBF (cf. Sec. 4.6) to create Intelligent Mentoring Bots (IMBots) [NLK\*20], chatbots tailored especially for mentoring processes. They integrate into common messaging platforms and provide just-in-time feedback. These IMBots are trained with the OSS RASA NLU<sup>5</sup>. Additionally, a *Mentoring Workbench* [RAZS20] integrates back into the respective LMS, providing the mentees with both the possibility to use the IMBot from within their known environment, as well as giving them feedback on their performance. We use an OIDC server to access all services in a modern and secure fashion. By coupling the OIDC identity and end-to-end encryption, the need for a central instance through which the communication takes place is eliminated.

<sup>5</sup><https://rasa.com/>

### 5.3.3 Summary and Outlook

In this ongoing demonstration of the approaches developed during this dissertation, we integrate our decentralized CIS infrastructure within a large heterogeneous technical context, based on OSS standards. The cluster has been set up as a public infrastructure and every external entity is able to connect to it. Data is currently coming from several German universities in the scope of the tech4comp project and from a MOOC supported by an ERASMUS+ project in the field of augmented reality. Our infrastructure supports both mentor and mentee in various organizations, following different legislative and organizational procedures, different LMS, as well as diverse target groups. Our development process, based on OSS commitment, allows us to quickly react on changing user requirements and organizational restrictions. Given the early status of the data processing, with first data sets successfully evaluated [NLK\*20], we provide a solid basis for future work.

## 5.4 Discussion and Conclusion

In this chapter, we applied and evaluated the methodologies and tool support developed during the course of this dissertation. Driven by our third research question “How can learning communities benefit from using a decentralized CIS?”, our real-world demonstrations and evaluations provided insights into the applicability and usefulness of scaffolding and operating decentralized CIS infrastructures and applications. The *Virtus Virtual VET Center* provided insights into the applicability of the MobSOS concept and MDWE techniques for course design of vocational learning programs. The *Distributed Noracle* application, along with its various artifacts developed alongside, most prominently the *Noracle Bot*, provided evidence that a blockchain-backed decentralized infrastructure can be applied in a distributed CoP. The self-managed infrastructure provided applicability in a scenario, in which previously community members would have used SNS and related tools to collaborate, ultimately relying on face-to-face meetings to apply their developed techniques for collaborative knowledge building and sharing. Finally, our third demonstration showed the integration and synergy effects of the artifacts developed within this dissertation with a larger infrastructure, based on state-of-the-art OSS industry standards. The very heterogeneous software landscape within the project is merged seamlessly with the decentralized CIS presented in this dissertation, showing the general applicability of our infrastructure within larger contexts.





# Chapter 6

## Conclusion and Future Work

This chapter sums up and discusses the contents of this dissertation with respect to the posed research questions. We then identify future work and provide an outlook on the developments of decentralized CIS for lifelong learning communities.

### 6.1 Conclusion

In this dissertation, we tried to answer the question of how *Decentralized Community Information Systems* can support *Communities of Practice*. Rooted in centralized CIS research [Klam10], taking into account findings from CIS success awareness [Renz16], NRT collaborative modeling [Nico18] and requirements elicitation [Kore20], we drive this research stream forward with a decentralized and distributed perspective on CIS for lifelong learning communities.

Chapter 3 introduced the technical underpinnings to support CoPs with the means to host their own, decentralized Web information system infrastructure. This way, a CoP regains control over their data, while being able to scale the infrastructure according to their needs. Following the design science principles, we produced several artifacts to provide possible answers to the question of what properties a decentralized, self-hosted infrastructure for CIS needs to fulfill. Our microservice discovery mechanism introduces an API metadata-based approach to provide communities with information on available services they could make use of in their community application (development). Decentralizing this approach by introducing las2peer's accompanying blockchain, we propose a possible solution for making this knowledge – which services are running on which nodes, what functionality do they provide, and how could they be used in ones own community

– accessible in a secure and verifiable way. This shift of the data being stored in centralized community’s metadata repositories [Klam10] to a decentralized microservice infrastructure, hosted by community members, opens up possibilities for a more democratic and egalitarian management of community knowledge. This self-hosted, decentralized infrastructure relies on the willingness of participants to contribute. By incentivizing community members to participate in the hosting of their community’s infrastructure, the burden of hosting a CIS can be distributed among the whole community. As one possible solution to this, we proposed the community contribution incentivization mechanism, based on reimbursing community members for their infrastructure and service development contributions. Finally, we present the blockchain-based verification and consent-management system for LA data. Thereby, we extend the decentralized knowledge infrastructure with means to connect to external, multimodal LA data sources, like sensors and LMS.

Chapter 4 then introduced means to support CoPs in creating their decentralized CIS. With the Community Application Editor, communities are provided with both a methodology and tool support for requirements elicitation, wireframing, modeling and coding their community applications – all on the Web, all in NRT. After deployment, we provide the NRT evaluation center for continuous service success evaluation. At any time, the community has the ability to revisit and modify their applications, the CAE supports the community throughout the complete community application life cycle. The second contribution we presented in Chapter 4 was the Social Bot Framework. It enables communities to create their own bots, similarly to the methodology the CAE introduced, by using a model-driven approach that reflects a complete social bot life cycle. By making use of conversational interfaces, the SBF provides additional ways of interacting with community applications.

Chapter 5 provided three demonstrations and evaluations of the infrastructure and scaffolding support tools introduced in this dissertation. We presented the V3C as a model-driven way of creating VET courses for PLEs. Evaluated on a large European scale with both trainers and students, we provided an in-depth evaluation of both model-driven principles and the NRT evaluation center. The Distributed Noracle, as an example of an infrastructure for community knowledge building, evaluated various aspects of the decentralized infrastructure introduced in Chapter 3. Finally the distributed mentoring infrastructure marks the third demonstration of our developed artifacts. This integration of a decentralized infrastructure with centralized, more “common” cloud-based system shows the flexible application of the solutions presented in this dissertation.

To sum up this dissertation, we have made the following contributions, sorted along the lines of our three guiding research questions.

**RQ 1 - Decentralized Infrastructures: What properties does a decentralized, self-hosted infrastructure for CIS need to fulfill?** Our main conceptual and technical basis was the *las2peer framework*, our decentralized, self-hosted CIS infrastructure [KRLJ16] with its *accompanying blockchain*. On top of it, we created three main artifacts:

- The *Decentralized Service Registry* [LJK119],
- a *Community Contribution Incentivation* mechanism [LSK120]
- and the *Verification and Consent Management of LA Data* [LBNK21].

**RQ 2 - Scaffolding: How to support the creation of decentralized CIS with the help of Web-based tool support?** Our contributions to answer this research question base on the Community Application Editor [LNKK16, LND\*16, LNKJ17, LNNK20] as both our technical underpinning as well as our conceptual backbone, while we extended and improved it with the following contributions:

- The *Live Code Editing* [LNWK18],
- the *Role-based Project Management*,
- a *NRT Evaluation Center* [LNNK18],
- its *Wireframing Support* [LNRK19, LNNK20]
- and the derived *Social Bot Framework* [NLK119, NLK\*20].

**RQ 3 - Learning: How can learning communities benefit from using a decentralized CIS?** Finally, we integrated our approaches and developed artifacts in various (informal) learning communities to observe their effects in practice. This lead to the development of three larger artifacts, created from the artifacts of both the decentralized infrastructure and the scaffolding support:

- The *Virtus Virtual VET Center* [LNK117, LNNK18],
- our *Infrastructure for Knowledge Building* [LFGK17, LGFK18, LGF\*20]
- and the ongoing *Distributed Mentoring Architecture* [KLN\*20, NLK\*20].

## 6.2 Future Work and Outlook

In this dissertation, we have made a case for a decentralized take on CIS. While we have shown in multiple applications the feasibility of this approach, this work only covers the first steps towards a fully decentralized knowledge management for CoPs. We identified several possibilities for future work, building on the outcomes of this dissertation, which we want to discuss in the following.

In line with the ongoing work of the tech4comp project, the efforts of extending the distributed mentoring infrastructure will most likely lead to a tighter integration of the tools and approaches developed in the course of this dissertation with state-of-the-art industrial cloud infrastructures. The already existing integration of complete las2peer networks within kubernetes clusters, developed within the final stages of this dissertation, will likely shape the development practices of las2peer more towards enterprise-grade software development and delivery processes. This is in-line with the goal of creating not only a platform that allows for decentralized, secure self-hosting of community knowledge, but also providing communities with an approach that allows *all* members – technical and non-technical – to create their applications in the way most suited to their expertise. As a side-effect, but equally as important, this research strand will improve the compatibility of our networks with software developed outside of these practices, the capabilities of the SBF to call external APIs is already a first step in this direction.

As a consequence, the scaffolding support of creating these community applications has to be tightly integrated within these new development scenarios. While the project- and dependency management of the CAE is a first step towards this, current research seeks ways to extend the support for more cluster-based deployment and management options. Additionally, the meta-models of CAE have to be constantly evolved to reflect current trends in Web engineering, which goes along with the challenges to adapt the wireframing support to do the same. An area worth investigating here is the integration of IFML, which would provide – if done completely – a life cycle reflection of the application, already on the level of the metamodel. Current research in this domain shows promising results [KKJa20], and an adaption of these principles to a complete scaffolding environment like the CAE could open up a new level of productivity of MDWE applied by CoPs.

With the rise of trends like the “Low Code / No Code” movement, the idea of non-technical people conducting programming tasks gains traction. The obvious reason of the lack of developers with an ever increasing demand for custom-tailored software leaves the industry little choice but to accept the fact that “There just aren’t

enough developers to go round.”<sup>1</sup> and extend software creation processes beyond trained developers. With this dissertation, we acknowledged this fact and with the CAE and the SBF, we provide both methodologies and tool support to include all stakeholders into the development processes.

Taking a step back and looking at the overall picture of current trends and developments, we see this work well embedded within the recent trend of decentralization. As so nicely put in [Mont16], “centralization and decentralization fight an eternal battle in many fields of the human culture”. While computing devices came a long way from centralized mainframes to a more decentralized “personal computing” approach, and now back to a more centralized approach with trends like cloud-gaming with thin-clients<sup>2</sup>, the Web tells its own story of this battle. While starting technically as a purely decentralized infrastructure of loosely connected nodes, information-wise it started as a place where only technical-savvy people were able to raise their voice. We then experienced the rise of the Web 2.0, which offered expressiveness power for all users, albeit at the cost of shifting data ownership to centralized cloud storage providers and SNS giants. P2P approaches during that time were only used for shady file-sharing applications, and DHT-based approaches never made it past the academic frontier.

However recently, this slowly seems to change. With the rise of blockchain technology, largely driven by the public recognition of cryptocurrencies being here to stay, there is a new case for decentralizing the Web once more. Projects like IPFS and Solid<sup>3</sup> gained traction in the academic- and lately also the developer world, while projects like Holochain<sup>4</sup>, with their connection to the cryptocurrency world, also connect to mainstream adaptation. They all share the idea of a decentralized ecosystem of services, empowering communities to regain control over their data once more, using P2P principles like blockchains and DHTs. With this dissertation, we share and contributed to this vision. Last but not least, with recent publications of very similar focus [HaCa19] and the recently started blockchain-based verification project MyEduLife<sup>5</sup> at the chair, we are confident that the ideas and vision of this research will be taken further and beyond.

---

<sup>1</sup>[https://thoughtworks.com/de/insights/articles/making\\_case\\_low-code\\_platforms](https://thoughtworks.com/de/insights/articles/making_case_low-code_platforms)

<sup>2</sup><https://stadia.google.com/>

<sup>3</sup><https://solidproject.org/>

<sup>4</sup><https://holochain.org/>

<sup>5</sup><https://bmbf.de/de/karliczek-bmbf-gestaltet-den-digitalen-weiterbildungsraum-13828.html>



# Bibliography

- [AABe10] Jonathan Arnowitz, Michael Arent, and Nevin Berger. *Effective Prototyping for Software Makers*. Elsevier Science, 2010.
- [ABC\*16] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A System for Large-Scale Machine Learning. In *Operating Systems Design and Implementation (OSDI 2016)*, pages 265–283. USENIX Association, 2016.
- [ABH\*02] Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David Martin, Drew McDermott, Sheila A. McIlraith, Srinu Narayanan, Massimo Paolucci, Terry Payne, and Katia Sycara. DAML-S: Web Service Description for the Semantic Web. In *The Semantic Web (SWC 2002)*, volume 2342 of *LNCS*, pages 348–363. Springer, 2002.
- [AFM\*05] Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. Web Service Semantics – WSDL-S. Technical report, W3C, 2005.
- [AHWa07] Fahad Aijaz, Bilal Hameed, and Bernhard Walke. Towards Peer-to-Peer Long-Lived Mobile Web Services. In *Innovations in Information Technologies (IIT 2007)*, pages 571–575. IEEE, 2007.
- [Ande06] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

- [Ange16] Michele Angelaccio. MetaPage: A Data Intensive MockupDD for Agile Web Engineering. In *Web Information Systems and Technologies (WEBIST 2016)*, volume 292 of *LNBIP*, pages 315–317, 2016.
- [BeAd10] Behram Beldagli and Tufan Adiguzel. Illustrating an Ideal Adaptive e-Learning: A Conceptual Framework. *Procedia-Social and Behavioral Sciences*, 2(2):5755–5761, 2010.
- [BLer17] Babar, Zia and Lapouchnian, Alexei and Yu, Eric SK. Chatbot Design - Towards a Social Analysis Using i\* and Process Architecture. *iStar*, 17:73–78, 2017.
- [Bayn15] Sian Bayne. Teacherbot: Interventions in Automated Teaching. *Teaching in Higher Education*, 20(4):455–467, 2015.
- [BBMR10] Geoffrey C. Bowker, Karen Baker, Florence Millerand, and David Ribes. Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment. In *International Handbook of Internet Research*, pages 97–117. Springer, 2010.
- [BCSS99] Guruduth Banavar, Tushar Chandra, Robert Strom, and Daniel Sturman. A Case for Message Oriented Middleware. In *Distributed Computing 1999*, volume 1693 of *LNCS*, pages 1–18. Springer, 1999.
- [BuDa09] Sonja Buchegger and Anwitaman Datta. A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges. In *Wireless On-Demand Network Systems and Services (WONS 2009)*, pages 161–168. IEEE, 2009.
- [Bene14] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System, 2014.
- [Bens16] Jan Benscheid. *3D Annotation for Collaborative Storytelling in Education and Research*. Bachelor Thesis, RWTH Aachen University, 2016.
- [Beng20] Lennart Bengtson. *Blockchain-based Verification of Learning Analytics Data*. Bachelor Thesis, RWTH Aachen University, 2020.
- [BrFr14] Marco Brambilla and Piero Fraternali. *Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML*. The MK/OMG Press. Morgan Kaufmann, 2014.



- [BHJa15] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Migrating to Cloud-Native Architectures Using Microservices: An Experience Report. In *Service-Oriented and Cloud Computing (ESOCC 2015)*, volume 567 of *CCIS*, pages 201–215. Springer, 2015.
- [BKCr18] Aras Bozkurt, Whitney Kilgore, and Matt Crosslin. Bot-Teachers in Hybrid Massive Open Online Courses (MOOCs): A Post-humanist Experience. *Australasian Journal of Educational Technology*, 34(3), 2018.
- [Bern89] Tim Berners-Lee. Information Management: A Proposal. No. CERN-DD-89-001-OC.
- [BHLa01] Tim Berners-Lee, James A. Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):28–37, 2001.
- [BrMi07] Peter Brusilovsky and Eva Millán. User Models for Adaptive Hypermedia and Adaptive Educational Systems. In *The Adaptive Web*, volume 4321 of *LNCS*, pages 3–53. Springer, 2007.
- [BaMa14] Ozalp Babaoglu and Moreno Marzolla. The People’s Cloud. *IEEE Spectrum*, 51(10):50–55, 2014.
- [BrPa98] Sergey Brin and Lawrence Page. The Anatomy of a large-scale hypertextual Web Search Engine. In *Computer Networks and ISDN Systems*, pages 107–117, 1998.
- [BrPe03] Peter Brusilovsky and Christoph Peylo. Adaptive and Intelligent Web-based Educational Systems. *International Journal of AI in Education*, 13:156–169, 2003.
- [BaPo17] Massimo Bartoletti and Livio Pompianu. An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns. In *Financial Cryptography and Data Security (FC 2017)*, volume 10323 of *LNCS*, pages 494–509. Springer, 2017.
- [BeRe01] Christian Bettstetter and Christoph Renner. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. In *EUNICE 2000, 6th Open European Summer School*. University of Twente, 2001.

- [BRCS14] Alejandro Bogarín, Cristóbal Romero, Rebeca Cerezo, and Miguel Sánchez-Santillán. Clustering for Improving Educational Process Mining. In *Learning Analytics And Knowledge (LAK 2014)*, pages 11–15. ACM, 2014.
- [Broo96] John Brooke. SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*, volume 189, pages 189–194. Taylor & Francis, 1996.
- [Brun17] Adam Brunnmeier. *Release PLEase! - Setting Up an Application Store based on an Agile Release Methodology for Personal Learning Environments*. Bachelor Thesis, RWTH Aachen University, 2017.
- [Bute17] Vitalik Buterin. A Modest Proposal for Ethereum 2.0, 2017-11-01.
- [Butl01] Brian S. Butler. Membership Size, Communication Activity, and Sustainability: A Resource-Based Model of Online Social Structures. *Information Systems Research*, 12(4):346–362, 2001.
- [CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1, 2001.
- [Ceci18] Melisa Cecilia. *Adding Microservice Discovery Support to a MDWE Framework*. Master Thesis, RWTH Aachen University, 2018.
- [CFBo00] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): A Modeling Language for Designing Web sites. *Computer Networks*, 33(1):137–157, 2000.
- [CKJA07] Chatti, Mohamed Amine and Klamma, Ralf and Jarke, Matthias and Naeve, Ambjörn. The Web 2.0 Driven SECI Model Based Learning Process. In *International Conference on Advanced Learning Technologies (ICALT 2007)*, pages 780–782. IEEE, 2007.
- [Cohe09] Steve Cohen. H.R.3149 - Equal Employment for All Act, 2009.
- [Conr20] Aaron Conrardy. *Chat Assessments with Social Bots*. Bachelor Thesis, RWTH Aachen University, 2020.
- [ChVa05] Ran Cheng and Julita Vassileva. User Motivation and Persuasion Strategy for Peer-to-Peer Communities. In *Hawaii International Conference on System Sciences (HICSS 2005)*. IEEE, 2005.

- [DALE16] Robert Dale. The Return of the Chatbots. *Natural Language Engineering*, 22(05):811–817, 2016.
- [DiBr16] Vania Dimitrova and Paul Brna. From Interactive Open Learner Modelling to Intelligent Mentoring: STyLE-OLM and Beyond. *International Journal of AI in Education*, 26(1):332–349, 2016.
- [BSFM13] Ángel Del Blanco, Ángel Serrano, Manuel Freire, Iván Martínez-Ortiz, and Baltasar Fernández-Manjón. E-Learning Standards and Learning Analytics. Can Data Collection be improved by using Standard Data Models? In *Global Engineering Education Conference (EDUCON 2013)*, pages 1255–1261. IEEE, 2013.
- [DeDa04] Prashant Dewan and Partha Dasgupta. Pride: Peer-to-Peer Reputation Infrastructure for Decentralized Environments. In *International World Wide Web Conference - Alternate Track Papers & Posters - (WWW Alt. 2004)*, page 480. ACM, 2004.
- [DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of Space. In *Advances in Cryptology (CRYPTO 2015)*, volume 9216 of *LNCS*, pages 585–605. Springer, 2015.
- [DLTM18] Massimiliano Dibitonto, Katarzyna Leszczynska, Federica Tazzi, and Carlo M. Medaglia. Chatbot in a Campus Environment: Design of LiSA, a Virtual Assistant to Help Students in Their University Life. In *International Conference on Human-Computer Interaction (HCI 2018)*, volume 10903 of *LNCS*, pages 103–116. Springer, 2018.
- [DaMa09] Wang Dan and Li Maozeng. A range query model based on DHT in P2P system. In *Networks Security, Wireless Communications and Trusted Computing (NSWCTC 2009)*, volume 1, pages 670–674. IEEE, 2009.
- [DwNa93] Cynthia Dwork and Moni Naor. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology (CRYPTO 1992)*, volume 740 of *LNCS*, pages 139–147. Springer, 1993.
- [DNE\*15] Michael Derntl, Petru Nicolaescu, Stephan Erdtmann, Ralf Klamma, and Matthias Jarke. Near Real-Time Collaborative Conceptual Modeling on the Web. In *Conceptual Modeling (ER 2015)*, volume 9381 of *LCNS*, pages 344–357. Springer, 2015.

- [Doli20] Philipp Dolif. *Versioning and Access Management in Web-Based Collaborative Model-Driven Web Engineering*. Bachelor Thesis, RWTH Aachen University, 2020.
- [DPR\*04] David DeSteno, Richard E. Petty, Derek D. Rucker, Duane T. Wegener, and Julia Braverman. Discrete Emotions and Persuasion: The Role of Emotion-induced Expectancies. *Journal of Personality and Social Psychology*, 86(1):43, 2004.
- [DoTa03] Martin Dougiamas and Peter Taylor. Moodle: Using Learning Communities to Create an Open Source Course Management System. In *Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA 2003)*, pages 171–178. Association for the Advancement of Computing in Education, 2003.
- [DeYu14] Li Deng and Dong Yu. Deep Learning: Methods and Applications. *Foundation and Trends in Signal Processing*, 7(3-4):197–387, 2014.
- [EFGK03] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.
- [Esha07] Haim Eshach. Bridging In-School and Out-of-School Learning: Formal, Non-Formal, and Informal Education. *Journal of Science Education and Technology*, 16(2):171–190, 2007.
- [Farm14] Randy Farmer. Web Reputation Systems and the Real World, 2014.
- [FoBr05] Andrea Forte and Amy Bruckman. Why Do People Write for Wikipedia? Incentives to Contribute to Open-Content Publishing. In *Sustaining Community – Incentive Mechanisms in Online Systems*, 2005.
- [FrCa06] Luke Fryer and Rollo Carpenter. Bots as Language Learning Tools. *Language Learning & Technology*, 10(3):8–14, 2006.
- [Ferg12] Rebecca Ferguson. Learning Analytics: Drivers, Developments and Challenges. *International Journal of Technology Enhanced Learning*, 4(5-6):304–317, 2012.
- [FGMi16] Tracie Farrell-Frey, George Gkotsis, and Alexander Mikroyannidis. Are you thinking what I’m thinking? Representing Metacognition with

- Question-based Dialogue. In *Workshop on Awareness and Reflection in Technology Enhanced Learning (ARTEL 2016)*, volume 1736 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.
- [Fiel00] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation, University of California, Irvine, 2000.
- [Fiel08] Roy T. Fielding. REST APIs Must Be Hypertext-Driven, 2008.
- [FiLa16] Michael Fischer and Monica Lam. From Books to Bots: Using Medical Literature to Create a Chat Bot. In *Workshop on IoT-enabled Healthcare and Wellness Technologies and Systems (IoT of Health 2016)*, pages 23–28. ACM, 2016.
- [FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Electronic Commerce (EC 2004)*, pages 102–111. ACM, 2004.
- [Fond17] Fondo Formación Euskadi S.L.L. Virtus Evaluation Report First Round, 2017.
- [Fowl10] Martin Fowler. Richardson Maturity Model: Steps toward the glory of REST, 2010.
- [FVD\*16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The Rise of Social Bots. *Communications of the ACM*, 59(7):96–104, 2016.
- [GAS\*16] Barbara Guidi, Tobias Amft, Andrea de Salve, Kalman Graffi, and Laura Ricci. DiDuSoNet: A P2P Architecture for Distributed Dunbar-based Social Networks. *Peer-to-Peer Networking and Applications*, 9(6):1177–1194, 2016.
- [GCHO05] Arthur C. Graesser, Patrick Chipman, Brian C. Haynes, and Andrew Olney. AutoTutor: An Intelligent Tutoring System With Mixed-Initiative Dialogue. *IEEE Transactions on Education*, 48(4):612–618, 2005.
- [GJK\*01] Günter Gans, Matthias Jarke, Stefanie Kethers, Gerhard Lakemeyer, Lutz Ellrich, Christiane Funken, and Martin Meister. Requirements Modelling for Organization Networks - a (Dis-)Trust-based Approach. In

- IEEE International Requirements Engineering Conference (RE 2001)*, pages 154–163. IEEE, 2001.
- [GME\*15] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-Centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.
- [GBGr16] Álvaro García-Recuero, Jeffrey Burdges, and Christian Grothoff. Privacy-Preserving Abuse Detection in Future Decentralised Online Social Networks. In *Data Privacy Management and Security Assurance (DPM 2016)*, volume 9963 of *LNCS*, pages 78–93. Springer, 2016.
- [Gutt99] Erik Guttman. Service Location Protocol: Automatic Discovery of IP Network Services. *IEEE Internet Computing*, 3(4):71–80, 1999.
- [Hank99] Sabine Hanke. The Performance of Concurrent Red-Black Tree Algorithms. In *Algorithm Engineering (WAE 1999)*, volume 1668 of *LNCS*, pages 286–300. Springer, 1999.
- [Harm06] Mark van Harmelen. Personal Learning Environments. In *International Conference on Advanced Learning Technologies (ICALT 2006)*, pages 815–816. IEEE, 2006.
- [HoCh11] Ilyoo B. Hong and Hwihyung Cho. The Impact of Consumer Trust on Attitudinal Loyalty and Purchase Intentions in B2C e-Marketplaces: Intermediary Trust vs. Seller Trust. *International Journal of Information Management*, 31(5):469–479, 2011.
- [HaCa19] Kanisius Kenneth Halim and Muhammad Zuhri Catur Candra. Dicey: A Blockchain Based Decentralized Service Registry. In *International Conference on Data and Software Engineering (ICoDSE 2019)*, pages 1–6. IEEE, 2019.
- [HaGi05] Jennifer Hammond and Pauline Gibbons. What is Scaffolding. *Teachers’ Voices*, 8:8–16, 2005.
- [HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

- [Hodg14] Bert H. Hodges. Rethinking Conformity and Imitation: Divergence, Convergence, and Social Understanding. *Frontiers in Psychology*, 5:726, 2014.
- [Hoss19] Philipp Hossner. *Integrating End Users Into Service Success Evaluation Processes*. Master Thesis, RWTH Aachen University, 2019.
- [HTWe16] Florian Hawlitschek, Timm Teubner, and Christof Weinhardt. Trust in the Sharing Economy. *Die Unternehmung*, 70(1):26–44, 2016.
- [HoWa15] Robert R. Hoffman and Paul Ward. Mentoring: A Leverage Point for Intelligent Systems? *IEEE Intelligent Systems*, 30(5):78–84, 2015.
- [IQLB16] Luca Iandoli, Ivana Quinto, Anna de Liddo, and Simon Buckingham Shum. On Online Collaboration and Construction of Shared Knowledge: Assessing Mediation Capability in Computer Supported Argument Visualization Tools. *Journal of the Association for Information Science and Technology*, 67(5):1052–1067, 2016.
- [Jans19] Tom Janson. *Decentralized Service Registry and Discovery in P2P Networks with Blockchain Technology*. Master Thesis, RWTH Aachen University, 2018.
- [JaJu99] Markus Jakobsson and Ari Juels. Proofs of Work and Bread Pudding Protocols. In *Secure Information Networks (CMS 1999)*, volume 23 of *IFIP*, pages 258–272. Springer, 1999.
- [JMRy11] Charles M. Judd, Gary H. McClelland, and Carey S. Ryan. *Data Analysis: A Model Comparison Approach*. Routledge, 2011.
- [KEBu08] Alice Kerly, Richard Ellis, and Susan Bull. CALMsystem: A Conversational Agent for Learner Modelling. In *Applications and Innovations in Intelligent Systems (SGAI 2007)*, pages 89–102. Springer, 2008.
- [Kent02] Stuart Kent. Model-Driven Engineering. In *Integrated Formal Methods (IFM 2002)*, volume 2335 of *LNCS*, pages 286–298. Springer, 2002.
- [KHBu07] Alice Kerly, Phil Hall, and Susan Bull. Bringing Chatbots into Education: Towards Natural Language Negotiation of Open Learner Models. *Knowledge-Based Systems*, 20(2):177–185, 2007.

- [Kim14] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751. Association for Computational Linguistics, 2014.
- [KoKr02] Nora Koch and Andreas Kraus. The Expressive Power of UML-based Web Engineering. In *International Workshop on Web-oriented Software Technology (IWWOST 2002)*, pages 105–119, 2002.
- [KKJa20] István Koren, Ralf Klamma, and Matthias Jarke. Direwolf Model Academy: An Extensible Collaborative Modeling Framework on the Web. In *Modellierung 2020 Workshops*, volume 2542 of *CEUR Workshop Proceedings*, pages 213–216. CEUR-WS.org, 2020.
- [Klam10] Ralf Klamma. *Social Software and Community Information Systems*. Habilitation, RWTH Aachen University, 2010.
- [KLN\*20] Ralf Klamma, Peter de Lange, Alexander Tobias Neumann, Benedikt Hensen, Milos Kravcik, Xia Wang, and Jakub Kuzilek. Scaling Mentoring Support with Distributed Artificial Intelligence. In *Intelligent Tutoring Systems (ITS 2020)*, volume 12149 of *LNCs*, pages 38–44. Springer, 2020.
- [Klus08] Matthias Klusch. Semantic Web Service Coordination. In *CASCOM: Intelligent Service Coordination in the Semantic Web*, pages 59–104. Birkhäuser Basel, 2008.
- [KMM\*08] Nora Koch, Santiago Meliá-Beigbeder, Nathalie Moreno-Vergara, Vicente Pelechano-Ferragud, Fernando Sánchez-Figueroa, and Juan-Manuel Vara-Mesa. Model-driven Web Engineering. *Upgrade-Novática Journal*, IX(2):40–45, 2008.
- [KMPe17] Peter M. Krafft, Michael Macy, and Alex Sandy Pentland. Bots as Virtual Confederates. In *Computer Supported Cooperative Work and Social Computing (CSCW 2017)*, pages 183–190. ACM, 2017.
- [KiNa12] Sunny King and Scott Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake: PeerCoin Whitepaper, 2012.
- [Kore20] István Koren. *DevOpsUse: Community-driven Continuous Innovation of Web Information Infrastructures*. Dissertation, RWTH Aachen University, 2020.



- [KeRy16] Jonathan M. Kevan and Paul R. Ryan. Experience API: Flexible, Decentralized and Activity-Centric Data Collection. *Technology, Knowledge and Learning*, 21(1):143–149, 2016.
- [KRLJ16] Ralf Klamma, Dominik Renzel, Peter de Lange, and Holger Janßen. las2peer – A Primer, 2016.
- [KSGa03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *International Conference on World Wide Web (WWW 2003)*, pages 640–651. ACM, 2003.
- [KSIg19] Milos Kravčik, Katharina Schmid, and Christoph Igel. Towards Requirements for Intelligent Mentoring Systems. In *International Workshop on Personalization and Recommendation on the Web and Beyond (ABIS 2019)*, pages 19–21. ACM, 2019.
- [KoHe15] Mark Kusters and Jeroen van der Heijden. From Mechanism to Virtue: Evaluating Nudge Theory. *Evaluation*, 21(3):276–291, 2015.
- [Lank13] Marc Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. The Enterprise Engineering Series. Springer, 3rd edition, 2013.
- [LBNK21] Peter de Lange, Lennart Bengtson, Alexander Tobias Neumann, and Ralf Klamma. Blockchain-based Data Verification and Consent Management for Trusted Learning Analytics Using Mentoring Chatbots. In *Learning Analytics & Knowledge (LAK 2021)*, 2021.
- [LFGK17] Peter de Lange, Tracie Farrell-Frey, Bernhard Göschlberger, and Ralf Klamma. Transferring a Question-Based Dialog Framework to a Distributed Architecture. In *Data Driven Approaches in Digital Education (EC-TEL 2017)*, volume 10474 of *LNCIS*, pages 549–552. Springer, 2017.
- [LGF\*20] Peter de Lange, Bernhard Göschlberger, Tracie Farrell, Alexander Tobias Neumann, and Ralf Klamma. Decentralized Learning Infrastructures for Community Knowledge Building. *Transactions on Learning Technologies*, 13(3):516 – 529, 2020.
- [LGFK18] Peter de Lange, Bernhard Göschlberger, Tracie Farrell, and Ralf Klamma. A Microservice Infrastructure for Distributed Communities

- of Practice. In *Lifelong Technology-Enhanced Learning (EC-TEL 2018)*, volume 11082 of *LNCS*, pages 172–186. Springer, 2018.
- [LJK119] Peter de Lange, Tom Janson, and Ralf Klamma. Decentralized Service Registry and Discovery in P2P Networks Using Blockchain Technology. In *Web Engineering (ICWE 2019)*, volume 11496 of *LNCS*, pages 296–311. Springer, 2019.
- [LND\*16] Peter de Lange, Petru Nicolaescu, Michael Derntl, Matthias Jarke, and Ralf Klamma. Community Application Editor: Collaborative Near Real-Time Modeling and Composition of Microservice-based Web Applications. In *Modellierung 2016 Workshop Proceedings*, pages 123–127, 2016.
- [LNK117] Peter de Lange, Petru Nicolaescu, and Ralf Klamma. VIRTUS Virtual VET Centre (V3C): A Learning Platform for Virtual Vocational Education and Training. In *Data Driven Approaches in Digital Education (EC-TEL 2017)*, volume 10474 of *LNCS*, pages 500–503. Springer, 2017.
- [LNKJ17] Peter de Lange, Petru Nicolaescu, Ralf Klamma, and Matthias Jarke. Engineering Web Applications Using Real-Time Collaborative Modeling. In *Collaboration and Technology (CRIWG 2017)*, volume 10391 of *LNCS*, pages 213–228. Springer, 2017.
- [LNKK16] Peter de Lange, Petru Nicolaescu, Ralf Klamma, and István Koren. DevOpsUse for Rapid Training of Agile Practices Within Undergraduate and Startup Communities. In *Adaptive and Adaptable Learning (EC-TEL 2016)*, volume 9891 of *LNCS*, pages 570–574. Springer, 2016.
- [LNNK18] Peter de Lange, Alexander Tobias Neumann, Petru Nicolaescu, and Ralf Klamma. An Integrated Learning Analytics Approach for Virtual Vocational Training Centers. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5(2):32–38, 2018.
- [LNNK20] Peter de Lange, Petru Nicolaescu, Alexander Tobias Neumann, and Ralf Klamma. Integrating Web-Based Collaborative Live Editing and Wireframing into a Model-Driven Web Engineering Process. *Data Science and Engineering*, 5(3):240–260, 2020.
- [LNRK19] Peter de Lange, Petru Nicolaescu, Mario Rosenstengel, and Ralf Klamma. Collaborative Wireframing for Model-Driven Web Engineering.

In *Web Information Systems Engineering (WISE 2019)*, volume 11881 of *LNCS*, pages 373–388. Springer, 2019.

- [LNWK18] Peter de Lange, Petru Nicolaescu, Thomas Winkler, and Ralf Klamma. Enhancing Model-Driven Web Engineering with Collaborative Live Coding. In *Modellierung 2018*, 2018.
- [LSK120] Peter de Lange, Michał Ślupczyński, and Ralf Klamma. Incentivizing Contribution in Decentralized Community Information Systems. In *Companion Proceedings of the Web Conference 2020 (WWW 2020)*, pages 636–644. ACM, 2020.
- [LCM\*10] Annabel M. Latham, Keeley A. Crockett, David A. McLean, Bruce Edmonds, and Karen O’Shea. Oscar: An Intelligent Conversational Agent Tutor to Estimate Learning Styles. In *International Conference on Fuzzy Systems (FUZZ 2010)*, pages 1–8. IEEE, 2010.
- [LiGo16] Ser Ling Lim and Ong Sing Goh. Intelligent Conversational Bot for Massive Online Open Courses (MOOCs). *arXiv preprint arXiv:1601.07065*, 2016.
- [LLSW10] Christoph Lenzen, Thomas Locher, Philipp Sommer, and Roger Wattenhofer. Clock Synchronization: Open Problems in Theory and Practice. In *Theory and Practice of Computer Science (SOFSEM 2010)*, volume 5901 of *LNCS*, pages 61–70. Springer, 2010.
- [LPWi13] Karsten O. Lundqvist, Guy Pursey, and Shirley Williams. Design and Implementation of Conversational Agents for Harvesting Feedback in eLearning Systems. In *Scaling up Learning for Sustained Impact (ECTEL 2013)*, volume 8095 of *LNCS*, pages 617–618. Springer, 2013.
- [LSPe82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
- [LMC\*16] Giuseppe La Torre, Salvatore Monteleone, Marco Cavallo, Valeria D’Amico, and Vincenzo Catania. A Context-Aware Solution to Improve Web Service Discovery and User-Service Interaction. In *Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing*,

*Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, pages 180–187. IEEE, 2016.

- [LaWe91] Jean Lave and Etienne Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.
- [LYPe14] Christine Lotter, Jan A. Yow, and Thomas T. Peters. Building a Community of Practice Around Inquiry Instruction Through a Professional Development Program. *International Journal of Science and Mathematics Education*, 12(1):1–23, 2014.
- [LZHS14] Chune Li, Richong Zhang, Jinpeng Huai, and Hailong Sun. A Novel Approach for API Recommendation in Mashup Development. In *International Conference on Web Services (ICWS 2014)*, pages 289–296. IEEE, 2014.
- [Maid14] MaidSafe. MaidSafe.net Announces Project SAFE to the Community: SAFE Network Whitepaper. 2014.
- [MMSF13] Dana Movshovitz-Attias, Yair Movshovitz-Attias, Peter Steenkiste, and Christos Faloutsos. Analysis of the Reputation System and User Contributions on a Question Answering Website. In *Advances in Social Network Analysis and Mining (ASONAM 2013)*, pages 886–893. ACM, 2013.
- [Matt16] Juri Mattila. The Blockchain Phenomenon – The Disruptive Potential of Distributed Consensus Architectures. (38), 2016.
- [MBL\*09] Fernando A. Mikic, Juan C. Burguillo, Martin Llamas, Daniel A. Rodriguez, and Eduardo Rodriguez. CHARLIE: An AIML-based Chat-terbot Which Works as an Interface Among INES and Humans. In *European Association for Education in Electrical and Information Engineering (EAEIE 2009)*, pages 1–6. IEEE, 2009.
- [MCH\*19] Naja Mack, Robert Cummings, Earl Huff, Kinnis Gosha, and Juan Gilbert. Exploring the Needs and Preferences of Underrepresented Minority Students for an Intelligent Virtual Mentoring System. In *International Conference on Human-Computer Interaction (HCI 2019)*, volume 1088, pages 213–221. Springer, 2019.

- [MoDu88] Paul V. Mockapetris and Kevin J. Dunlap. Development of the Domain Name System. *Communication Architecture and Protocols (SIGCOMM 1988)*, 18(4):123–133, 1988.
- [Merk79] Ralph C. Merkle. Method of Providing Digital Signatures, 1982.
- [Meye03] Katrina A. Meyer. Face-to-Face Versus Threaded Discussions: The Role of Time and Higher-Order Thinking. *Journal of Asynchronous Learning Networks*, 7(3):55–65, 2003.
- [MiKi94] Paul Milgram and Fumio Kishino. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information and Systems*, E77-D(12):1321–1329, 1994.
- [Mont16] Alberto Montresor. Reflecting on the Past, Preparing for the Future: From Peer-to-Peer to Edge-centric Computing. In *International Conference on Distributed Computing Systems (ICDCS 2016)*, pages 22–23. IEEE, 2016.
- [MPM\*05] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *Semantic Web Services and Web Process Composition*, volume 3387 of *LNCS*, pages 26–42. Springer, 2005.
- [MPOR18] Clodagh McLoughlin, Kunal D. Patel, Tom O’Callaghan, and Scott Reeves. The Use of Virtual Communities of Practice to Improve Inter-professional Collaboration and Education: Findings from an Integrated Review. *Journal of Interprofessional Care*, 32(2):136–142, 2018.
- [Naka08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report, 2008.
- [NDK113] Petru Nicolaescu, Michael Derntl, and Ralf Klamma. Browser-based Collaborative Modeling in Near Real-Time. In *Networking, Applications and Worksharing (CollaborateCom 2013)*, pages 335–344. IEEE, 2013.
- [NDK115] Nicolae Nistor, Michael Derntl, and Ralf Klamma. Learning Analytics: Trends and Issues of the Empirical Research of the Years 2011-2014. In

*Design for Teaching and Learning in a Networked World (EC-TEL 2015)*, volume 9307 of *LNCS*, pages 453–459. Springer, 2015.

- [Neum18] Alexander Neumann. *Model-Driven Construction & Utilization of Social Bots for Technology Enhanced Learning*. Master Thesis, RWTH Aachen University, 2018.
- [Newm15] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O’Reilly, 2015.
- [Nico18] Petru Nicolaescu. *Engineering Web Community Information Systems via Near Real-Time Collaborative Modeling Support*. Dissertation, RWTH Aachen University, 2018.
- [NJDK16] Petru Nicolaescu, Kevin Jahns, Michael Derntl, and Ralf Klamma. Near Real-Time Peer-to-Peer Shared Editing on Extensible Data Types. In *International Conference on Supporting Group Work (GROUP 2016)*, pages 39–49. ACM, 2016.
- [NJM\*12] Shirin Nilizadeh, Sonia Jahid, Prateek Mittal, Nikita Borisov, and Apu Kapadia. Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching. In *International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2012)*, pages 337–348. ACM, 2012.
- [NLK119] Alexander Tobias Neumann, Peter de Lange, and Ralf Klamma. Collaborative Creation and Training of Social Bots in Learning Communities. In *Collaboration and Internet Computing (CIC 2019)*, pages 11–19. IEEE, 2019.
- [NLK\*20] Alexander Tobias Neumann, Peter de Lange, Ralf Klamma, Norbert Pengel, and Tamar Arndt. Intelligent Mentoring Bots in Learning Management Systems: Concepts, Realizations and Evaluations. In *Learning Technologies and Systems (ICWL 2020)*, volume 12511 of *LNCS*, pages 3–14. Springer, 2020.
- [NLKK19] Alexander Neumann, Peter de Lange, Michael Kretschmer, and Ralf Klamma. tech4comp Report zu AP 3.1: Technologiestudie einer Infrastruktur zur verteilten Datenanalyse.

- [NRD\*18] Petru Nicolaescu, Mario Rosenstengel, Michael Derntl, Ralf Klamma, and Matthias Jarke. Near Real-time Collaborative Modeling for View-based Web Information Systems Engineering. *Information Systems*, 74(1):23–39, 2018.
- [NoTa95] Ikujiro Nonaka and Hirotaka Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.
- [NWQ\*02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: a P2P Networking Infrastructure Based on RDF. In *International World Wide Web Conference (WWW 2002)*, pages 604–615. ACM, 2002.
- [Open18] OpenAPI Initiative. The OpenAPI Specification: Version 3.0.2, 2018.
- [ORei07] Tim O’Reilly. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, 65:17–37, 2007.
- [OvSt15] Jarrod Overson and Jason Strimpel. *Developing Web Components: UI From JQuery to Polymer*. O’Reilly Media, Inc., 2015.
- [OcTe17] Xavier Ochoa and Stefaan Ternier. Technical Learning Infrastructure, Interoperability and Standards. In *Technology Enhanced Learning*, pages 145–155. Springer, 2017.
- [PeDi18] Juanan Pereira and Oscar Díaz. A Quality Analysis of Facebook Messenger’s Most Popular Chatbots. In *Symposium on Applied Computing (SAC 2018)*, pages 2144–2150. ACM, 2018.
- [PRMa11] Kevin R. Page, David C. de Roure, and Kirk Martinez. REST and Linked Data: a Match Made for Domain Driven Development? In *International Workshop on RESTful Design (WS-REST 2011)*, pages 22–25. ACM, 2011.
- [PaLo04] Alexandros Paramythis and Susanne Loidl-Reisinger. Adaptive Learning Environments and e-Learning Standards. *Electronic Journal of e-Learning*, 2(1):181–194, 2004.

- [Prit08] Dan Pritchett. BASE: An ACID Alternative. *Queue*, 6(3):48–55, 2008.
- [PGLa18] Sara Perez-Soler, Esther Guerra, and Juan de Lara. Collaborative Modeling and Group Decision Making Using Chatbots in Social Networks. *IEEE Software*, 35(6):48–54, 2018.
- [PSLa80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM*, 27(2):228–234, 1980.
- [PSNS03] Massimo Paolucci, Katia P. Sycara, Takuya Nishimura, and Naveen Srinivasan. Using DAML-S for P2P Discovery. In *International Conference on Web Services (ICWS 2003)*, pages 203–207. CSREA Press, 2003.
- [PTRC07] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, 2007.
- [PZLe08] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In *International Conference on World Wide Web (WWW 2008)*, pages 805–814. ACM, 2008.
- [RANR03] Claudia Roda, Albert Angehrn, Thierry Nabeth, and Liana Razmerita. Using Conversational Agents to support the Adoption of Knowledge Sharing Practices. *Interacting with Computers*, 15(1):57–89, 2003.
- [Rast21] Erdzan Rastoder. *Web-based Cluster Deployment for a Large-Scale Distributed Infrastructure*. Bachelor Thesis, RWTH Aachen University, 2021.
- [RAZS20] Jana Riedel, Björn Adelberg, Julia Zawidzki, and Sylvia Schulze-Achatz. Creating an Infrastructure to Integrate Specialized Services to Proprietary LMS. In *ICERI 2020 Proceedings*, pages 1846–1854. IATED, 2020.
- [RBKJ13] Dominik Renzel, Malte Behrendt, Ralf Klamma, and Matthias Jarke. Requirements Bazaar: Social Requirements Engineering for Community-Driven Innovation. In *Requirements Engineering (RE 2013)*, pages 326–327. IEEE, 2013.



- [RoDr01] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *International Conference on Distributed Systems Platforms (Middleware 2001)*, volume 2218 of *LNCS*, pages 329–350. Springer, 2001.
- [Renz16] Dominik Renzel. *Information Systems Success Awareness for Professional Long Tail Communities of Practice*. Dissertation, RWTH Aachen University, 2016.
- [RGR\*12] José Matías Rivero, Julián Grigera, Gustavo Rossi, Esteban Robles Luna, and Nora Koch. Towards Agile Model-Driven Web Engineering. In *IS Olympics: Information Systems in a Diverse World: (CAiSE 2011)*, volume 107 of *LNBIP*, pages 142–155. Springer, 2012.
- [Rich15] Mark Richards. *Microservices vs. Service-Oriented Architecture*. O’Reilly Media, 2015.
- [Rick20] Julius Rickert. *Secure and User-friendly Authentication for a Decentralized Service Platform*. Bachelor Thesis, RWTH Aachen University, 2020.
- [RKJa15] Dominik Renzel, Ralf Klamma, and Matthias Jarke. IS Success Awareness in Community-Oriented Design Science Research. In *New Horizons in Design Science: Broadening the Research Agenda (DESRIST 2015)*, volume 9073 of *LNCS*, pages 413–420. Springer, 2015.
- [RKKN15] Dominik Renzel, Ralf Klamma, Miloš Kravčák, and Alexander Nussbaumer. Tracing Self-Regulated Learning in Responsive Open Learning Environments. In *Advances in Web-Based Learning (ICWL 2015)*, volume 9412 of *LNCS*, pages 155–164. Springer, 2015.
- [RKLb09] Michael Rambold, Holger Kasinger, Florian Lautenbacher, and Bernhard Bauer. Towards Autonomic Service Discovery A Survey and Comparison. In *International Conference on Services Computing (SCC 2009)*, pages 192–201. IEEE, 2009.
- [RKZF00] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation Systems. *Communications of the ACM*, 43(12):45–48, 2000.

- [RoLa96] Ronald L. Rivest and Butler Lampson. SDSI: A Simple Distributed Security Infrastructure. In *Advances in Cryptology (CRYPTO 1996)*, LNCS. Springer, 1996.
- [Rose17] Mario Rosenstengel. *A Near Real-Time Collaborative Wireframing Editor for Agile Model-Driven Web Engineering*. Master Thesis, RWTH Aachen University, 2017.
- [REB\*12] Peter van Rosmalen, Johan Eikelboom, Erik Bloemers, Kees van Winzum, and Pieter Spronck. Towards a Game-Chatbot: Extending the Interaction in Serious Games. In *European Conference on Games Based Learning (ECGBL 2012)*, pages 525–532. Academic Publishing International Limited, 2012.
- [Royt19] Philipp Roytburg. *A Multimodal Mentoring Cockpit for Tutor Support*. Bachelor Thesis, RWTH Aachen University, 2019.
- [RiRo13] José Matías Rivero and Gustavo Rossi. MockupDD: Facilitating Agile Support for Model-Driven Web Engineering. In *Current Trends in Web Engineering (ICWE 2013)*, volume 8295 of LNCS, pages 325–329. Springer, 2013.
- [RRG\*11] José Rivero, Gustavo Rossi, Julián Grigera, Esteban Robles Luna, and Antonio Navarro. From Interface Mockups to Web Application Models. In *International Conference on Web Information Systems Engineering*, pages 257–264, 2011.
- [RiSa12] Angelica Riskey and Marife Sanchez-Garcia. The Jury is still out: Psychoemotional Support in Peer e-Mentoring for Transition to University. *The Internet and Higher Education*, 15(3):213–221, 2012.
- [RSK112] Dominik Renzel, Patrick Schlebusch, and Ralf Klamma. Today’s Top “RESTful” Services and Why They Are Not RESTful. In *Web Information Systems Engineering (WISE 2012)*, volume 7651 of LNCS, pages 354–367. Springer, 2012.
- [RDVi14] Steven Raemaekers, Arie van Deursen, and Joost Visser. Semantic Versioning versus Breaking Changes: A Study of the Maven Repository. In *Source Code Analysis and Manipulation (SCAM 2014)*, pages 215–224. IEEE, 2014.

- [ScBe06] Marlene Scarlia and Carl Bereiter. Knowledge Building: Theory, Pedagogy, and Technology. *Cambridge Handbook of the Learning Sciences*, 2006.
- [SBSc18] Martin Schanzenbach, Georg Bamm, and Julian Schutte. reclaimID: Secure, Self-Sovereign Identities Using Name Systems and Attribute-Based Encryption. In *International Conference On Trust, Security And Privacy In Computing And Communications/International Conference On Big Data Science And Engineering (TrustCom/BigDataSE 2018)*, pages 946–957. IEEE, 2018.
- [Scar02] Marlene Scardamalia. Collective Cognitive Responsibility for the Advancement of Knowledge. *Liberal Education in a Knowledge Society*, 97:67–98, 2002.
- [Scha97] Roger C. Schank. *Virtual learning: A revolutionary approach to building a highly skilled workforce*. Irwin Professional Publishing, 1997.
- [Schu00] Daniel Schugurensky. *The Forms of Informal Learning: Towards a Conceptualization of the Field*. NALL Working Paper. Centre for the Study of Education and Work, 2000.
- [Schm06] Douglas C. Schmidt. Model-Driven Engineering. *Computer*, 39(2):25–31, 2006.
- [SGA\*05] Ozgur D. Sahin, Cagdas E. Gerede, Divyakant Agrawal, Amr El Abbadi, Oscar Ibarra, and Jianwen Su. SPiDeR: P2P-Based Web Service Discovery. In *Service-Oriented Computing (ICSOC 2005)*, volume 3826 of *LNCS*, pages 157–169. Springer, 2005.
- [Shaw12] Alan Shaw. Using Chatbots to Teach Socially Intelligent Computing Principles in Introductory Computer Science Courses. In *Information Technology - New Generations (ITNG 2012)*, pages 850–851. IEEE, 2012.
- [SJPr06] Satish Narayana Srirama, Matthias Jarke, and Wolfgang Prinz. Mobile Web Service Provisioning. In *Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW 2006)*, page 120. IEEE, 2006.
- [SmKo99] Marc A. Smith and Peter Kollock. *Communities in Cyberspace*. Routledge, 1999.

- [SKD\*14] José Luis Santos, Joris Klerkx, Erik Duval, David Gago, and Luis Rodríguez. Success, Activity and Drop-outs in MOOCs an Exploratory Study on the UNED COMA Courses. In *Learning Analytics and Knowledge (LAK 2014)*, pages 98–102. ACM, 2014.
- [SKWL99] Katia Sycara, Matthias Klusch, Seth Widoff, and Jianguo Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. *ACM SIGMOD Record*, 28(1):47–53, 1999.
- [SLKe04] Chenliang Sun, Yi Lin, and B. Kemme. Comparison of UDDI Registry Replication Strategies. In *International Conference on Web Services (ICWS 2004)*, pages 218–225. IEEE, 2004.
- [Sloa17] Paul Sloane. *The Leader’s Guide to Lateral Thinking Skills: Unlock the Creativity and Innovation in You and Your Team*. Kogan Page, 2017.
- [Slup20] Michal Slupczynski. *Incentivizing Community Contributions in Decentralized Information Systems*. Master Thesis, RWTH Aachen University, 2020.
- [ScPa04] Cristina Schmidt and Manish Parashar. A Peer-to-Peer Approach to Web Service Discovery. *World Wide Web*, 7(2):211–229, 2004.
- [ScRo98] Daniel Schwabe and Gustavo Rossi. An Object Oriented Approach to Web-based Applications Design. *TAPOS*, 4(4):207–225, 1998.
- [SSDN02] Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In *Peer-to-Peer Computing (P2P 2002)*, pages 104–111. IEEE, 2002.
- [SSG\*17] Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeshwar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. A Deep Reinforcement Learning Chatbot. *CoRR*, abs/1709.02349, 2017.
- [ShTo03] Valerie Shute and Brendon Towle. Adaptive E-Learning. *Educational Psychologist*, 38(2):105–114, 2003.

- [STKS15] Angel Suarez, Stefaan Ternier, Marco Kalz, and Marcus Specht. Supporting Inquiry-based Learning with Google Glass (GPIM). *Interaction Design and Architecture Journal*, 24:100–110, 2015.
- [TGDN18] Ramón Toala, Filipe Gonçalves, Dalila Durães, and Paulo Novais. Adaptive and Intelligent Mentoring to Increase User Attentiveness in Learning Activities. In *Advances in Artificial Intelligence (IBERAMIA 2018)*, volume 11238 of *LNAI*, pages 145–155. Springer, 2018.
- [TSNe03] Uwe Thaden, Wolf Siberski, and Wolfgang Nejdl. A Semantic Web based Peer-to-Peer Service Registry Network. Technical report, Learning Lab Lower Saxony, 2013.
- [UDDI00] UDDI Coalition. The UDDI Technical White Paper. Technical report, 2000.
- [VoBu15] Fabian Vogelsteller and Vitalik Buterin. EIP 20: ERC-20 Token Standard, 2015.
- [VaPi18] Enza Varghese and M. T. Rajappan Pillai. A Standalone Generative Conversational Interface Using Deep Learning. In *Information, Communication and Computing Technology (ICICCT 2018)*, volume 835 of *CCIS*, pages 1915–1920. Springer, 2018.
- [W3C13] W3C. W3C Data Activity: Building the Web of Data, 2013.
- [Wall03] Richard Wallace. *The Elements of AIML Style*, volume 139. 2003.
- [WBRo76] David Wood, Jerome S. Bruner, and Gail Ross. The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976.
- [Weiz66] Joseph Weizenbaum. ELIZA – A Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1):36–45, 1966.
- [Weng98] Etienne Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Learning in Doing. Cambridge University Press, 1998.
- [WHRo14] Jon Whittle, John Hutchinson, and Mark Rouncefield. The State of Practice in Model-Driven Engineering. *IEEE Software*, 31(3):79–85, 2014.

- [Wies19] Niels Wiessner. *Chat Interfaces for Social Bots in a Peer-to-Peer Environment*. Bachelor Thesis, RWTH Aachen University, 2019.
- [Wink16] Thomas Winkler. *A Live Collaborative Editing and Deployment Approach for Model-based Community Applications*. Bachelor Thesis, RWTH Aachen University, 2016.
- [Wood14] Gavin Wood. *Ethereum: A Secure Decentralized Transaction Ledger*. Technical report, 2014.
- [WiSo18] Rainer Winkler and Matthias Söllner. Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. In *Academy of Management Annual Meeting (AOM 2018)*, 2018.
- [WSGr14] Matthias Wachs, Martin Schanzenbach, and Christian Grothoff. A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System. In *Cryptology and Network Security (CANS 2014)*, volume 8813 of *LNCS*, pages 127–142. Springer, 2014.
- [WZNP13] Annika Wolff, Zdenek Zdrahal, Andriy Nikolov, and Michal Pantucek. Improving Retention: Predicting At-risk Students by Analysing Clicking Behaviour in a Virtual Learning Environment. In *Learning Analytics and Knowledge (LAK 2013)*, pages 145–149. ACM, 2013.
- [LiLi04] Li Xiong and Ling Liu. PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.
- [XhPo10] Fatos Xhafa and Alex Poulovassilis. Requirements for Distributed Event-Based Awareness in P2P Groupware Systems. In *Advanced Information Networking and Applications (AINA 2010)*, pages 220–225. IEEE, 2010.
- [YCCI16] Mengting Yan, Paul Castro, Perry Cheng, and Vatche Ishakian. Building a Chatbot with Serverless Computing. In *International Workshop on Mashups of Things and APIs (MOTA 2016)*, pages 1–4. ACM, 2016.
- [YaZh04] Feng Yan and Shouyi Zhan. A Peer-to-Peer Approach with Semantic Locality to Service Discovery. In *Grid and Cooperative Computing (GCC 2004)*, volume 3251 of *LNCS*, pages 831–834. Springer, 2004.

- [ZhLi13] Huanyu Zhao and Xiaolin Li. VectorTrust: Trust Vector Aggregation Scheme for Trust Management in Peer-to-Peer Networks. *The Journal of Supercomputing*, 64(3):805–829, 2013.





# List of Figures

|      |   |    |
|------|---|----|
| 1.1  | Overarching design science process of this dissertation. . . . .  | 5  |
| 1.2  | Overview on the artifacts produced in this dissertation. . . . .  | 6  |
| 2.1  | Venn diagram that shows the interplay between the three main research areas this dissertation is embedded in. . . . . | 10 |
| 3.1  | Iterations of the design science process we followed in Chapter 3. .  | 37 |
| 3.2  | The transformation of a monolithic CIS ([Klam10], left) to a decentralized CIS (right). . . . .                       | 38 |
| 3.3  | las2peer basic architecture [KRLJ16] (adapted). . . . .   | 39 |
| 3.4  | Metadata extraction from service model and metadata widget. . .   | 42 |
| 3.5  | Exemplary matching levels of two service endpoints. . . . .   | 43 |
| 3.6  | Screenshot of the metadata editor. . . . .  | 45 |
| 3.7  | Screenshot of the deployment viewer. . . . .  | 45 |
| 3.8  | Screenshot of the service matching viewer. . . . .  | 46 |
| 3.9  | Usage scenario without decentralized service registry. . . . .  | 49 |
| 3.10 | The registry smart contract data. . . . .   | 51 |
| 3.11 | Usage scenario with decentralized service registry. . . . .   | 52 |
| 3.12 | Architecture and information flow during common operations. . .   | 53 |
| 3.13 | Screenshot of the service explorer, depicting two applications currently available in the network. . . . .            | 54 |
| 3.14 | Delegated username registration with signed smart contract call. .  | 56 |
| 3.15 | Evaluation of the decentralized service registry: questionnaire responses. . . . .                                    | 58 |
| 3.16 | System overview of the incentivization system. . . . .  | 63 |
| 3.17 | Reputation dashboard landing page, showing a list of community members together with their reputation. . . . .        | 65 |

|      |   |     |
|------|---|-----|
| 3.18 | Service author and hoster reputation, integrated into the service explorer of the decentralized service registry. . . . .       | 66  |
| 3.19 | Reputation wallet view. . . . .   | 66  |
| 3.20 | Reputation payout confirmation dialog. . . . .  | 67  |
| 3.21 | Evaluation of the community incentivization system: questionnaire responses. . . . .  | 68  |
| 3.22 | Evaluation of the community incentivization system: For which reasons would you consider hosting a service for a community? . . | 69  |
| 3.23 | Usage scenario without verified LA data. . . . .  | 71  |
| 3.24 | Usage scenario with verified LA data. . . . .   | 73  |
| 3.25 | Architectural overview of the LA data verification system. . . . .  | 75  |
| 3.26 | Sample conversation with the LA verification bot. . . . .   | 77  |
| 3.27 | Evaluation of the verification system: questionnaire responses. . .   | 79  |
| 3.28 | All smart contracts developed in this chapter. . . . .  | 80  |
| 4.1  | Iterations of the design science process we followed in Chapter 4. .  | 85  |
| 4.2  | Overview of the MDWE approach. . . . .  | 89  |
| 4.3  | Different representations of the same frontend component. . . . .   | 90  |
| 4.4  | The underlying Web application metamodel used in the CAE. . . . .   | 91  |
| 4.5  | ER diagram of the CAE. . . . .  | 93  |
| 4.6  | Screenshot of the NRT Evaluation Center flow. . . . .   | 96  |
| 4.7  | Mapping of the SUIT- to the MDWE metamodel. . . . .   | 97  |
| 4.8  | Screenshot of the frontend component modeling space. . . . .  | 101 |
| 4.9  | Screenshot of the CAE project management features. . . . .  | 102 |
| 4.10 | Screenshot of the NRT Evaluation Center PWA. . . . .  | 103 |
| 4.11 | Screenshots of the versioning widget. . . . .   | 104 |
| 4.12 | Architecture of the CAE. . . . .  | 105 |
| 4.13 | Results of the wireframing user evaluation. . . . .   | 113 |
| 4.14 | Screenshot of a social bot model. . . . .   | 119 |
| 4.15 | Social bot life-cycle. . . . .  | 120 |
| 4.16 | Screenshot of the method browser, in this example depicting the available methods of the Distributed Noracle service. . . . .   | 122 |
| 4.17 | Screenshot of the Bot Forge Web interface. . . . .  | 124 |
| 4.18 | Results of the SBF user evaluation. . . . .   | 126 |
| 5.1  | Above: The modeling view of the V3C. Below: The resulting PLE.  | 135 |
| 5.2  | A course room of the V3C platform. . . . .  | 136 |

|      |   |     |
|------|---|-----|
| 5.3  | The basic course edit view, showing both the multi-language feature and the translation option in the learning-designer view. . . .   | 137 |
| 5.4  | V3C evaluation: Module activity over time. . . . .  | 142 |
| 5.5  | V3C evaluation: Activities compared to scored results. . . . .  | 143 |
| 5.6  | V3C evaluation: Assessment results without drop-outs. . . . .   | 143 |
| 5.7  | V3C evaluation: Drop-out rate. . . . .  | 144 |
| 5.8  | V3C evaluation: Questionnaire results. . . . .  | 145 |
| 5.9  | Infrastructure for Community Knowledge Building: Design science process. . . . .  | 149 |
| 5.10 | Screenshot of the Distributed Noracle application, showing a question-based dialog space used in one of our evaluations. . . .  | 151 |
| 5.11 | Exemplary usage scenario of the Distributed Noracle. . . . .  | 152 |
| 5.12 | A question creation process in the Distributed Noracle. . . . .   | 153 |
| 5.13 | An exemplary general bot message of the Distributed Noracle, as it was send to the evaluators during our pilot bot evaluation. . . .  | 155 |
| 5.14 | A personal bot message of the Distributed Noracle send to an evaluator during our pilot bot evaluation. . . . .   | 156 |
| 5.15 | Usage statistics of the one-year decentralized infrastructure deployment, continuously providing the Distributed Noracle application since our first real-world pedagogical usage evaluation. . . . | 163 |
| 5.16 | Visualization of a Reddit megathread as a Distributed Noracle space. 165  |     |
| 5.17 | Mentoring support infrastructure. . . . .   | 174 |



# List of Tables

- 4.1 Results of the wireframe activity evaluation for each session and participant. The relative time spend in each widget, as well as the absolute number of activities for each widget is given. . . . . 114
- 5.1 V3C evaluation participants per module. (SE = Social Entrepreneurship, TH = Tourism & Hospitality) . . . . . 140



# Appendix A

## List of Abbreviations

|  |    |
|--|----|
| <b>AIML</b> Artificial Intelligence Markup Language . . . . .    | 29 |
| <b>ALE</b> Adaptive Learning Environments . . . . .              | 25 |
| <b>API</b> Application Programming Interface . . . . .           | 14 |
| <b>CA</b> Certificate Authorities . . . . .                      | 23 |
| <b>CAE</b> Community Application Editor . . . . .                | 6  |
| <b>CITS</b> Conversational Intelligent Tutoring System . . . . . | 30 |
| <b>CIS</b> Community Information System . . . . .                | 1  |
| <b>CMS</b> Content Management System . . . . .                   | 1  |
| <b>CoI</b> Community of Inquiry . . . . .                        | 2  |

|   |     |
|---|-----|
| <b>CoP</b> Community of Practice . . . . .                                    | 1   |
| <b>CPU</b> Central Processing Unit . . . . .                                  | 58  |
| <b>CRDT</b> Conflict-free Replicated Data Type . . . . .                      | 105 |
| <b>DDD</b> Domain Driven Design . . . . .                                     | 11  |
| <b>DHT</b> Distributed Hash Table . . . . .                                   | 11  |
| <b>DNS</b> Domain Name Server . . . . .                                       | 23  |
| <b>DOM</b> Document Object Model . . . . .                                    | 12  |
| <b>ECQA</b> European Certification and Qualification Association . . . . .    | 136 |
| <b>ECVET</b> European Credit System for Vocational Education and Training . . | 133 |
| <b>EDA</b> Event-Driven Architecture . . . . .                                | 40  |
| <b>EVS</b> European Voluntary Service . . . . .                               | 7   |
| <b>EPM</b> Educational Process Mining . . . . .                               | 31  |
| <b>ER</b> Entity Relationship . . . . .                                       | 92  |
| <b>GUHA</b> General Unary Hypothesis Automaton . . . . .                      | 32  |



|  |     |
|--|-----|
|  | 217 |
| <b>HCI</b> Human-Computer Interaction . . . . .                            | 9   |
| <b>HTML</b> HyperText Markup Language . . . . .                            | 97  |
| <b>HTTP</b> HyperText Transfer Protocol . . . . .                          | 44  |
| <b>IBL</b> Inquiry-based Learning . . . . .                                | 33  |
| <b>IFML</b> Interaction Flow Modeling Language . . . . .                   | 27  |
| <b>IMBot</b> Intelligent Mentoring Bot . . . . .                           | 174 |
| <b>IMS</b> Intelligent Mentoring System . . . . .                          | 172 |
| <b>IFML</b> Interaction Flow Modeling Language . . . . .                   | 27  |
| <b>IPFS</b> InterPlanetary File System . . . . .                           | 11  |
| <b>ITS</b> Intelligent Tutoring Systems . . . . .                          | 172 |
| <b>IWC</b> Inter Widget Communication . . . . .                            | 91  |
| <b>JSON</b> JavaScript Object Notation . . . . .                           | 14  |
| <b>JTELSS</b> Joint European Summer School on Technology Enhanced Learning | 158 |
| <b>KPI</b> Key Performance Indicator . . . . .                             | 102 |

|  |     |
|--|-----|
| <b>LA</b> Learning Analytics . . . . .                                     | 7   |
| <b>LMS</b> Learning Management System . . . . .                            | 31  |
| <b>LRS</b> Learning Record Store . . . . .                                 | 31  |
| <b>MDD</b> Model Driven Development . . . . .                              | 24  |
| <b>MDWE</b> Model-Driven Web Engineering . . . . .                         | 3   |
| <b>MobSOS</b> Mobile Community Information System Oracle for Success . . . | 40  |
| <b>MobSOS QV</b> MobSOS Query Visualizer . . . . .                         | 140 |
| <b>MockupDD</b> Mockup Driven Development . . . . .                        | 28  |
| <b>MOM</b> Message Oriented Middleware . . . . .                           | 39  |
| <b>MOOC</b> Massive Open Online Course . . . . .                           | 7   |
| <b>NLU</b> Natural Language Understanding . . . . .                        | 29  |
| <b>NRT</b> Near Real-Time . . . . .  | 7   |
| <b>OAS</b> OpenAPI Specification . . . . .                                 | 14  |
| <b>OCL</b> Object Constraint Language . . . . .                            | 26  |

|  |     |
|--|-----|
|  | 219 |
| <b>OIDC</b> OpenID Connect . . . . .                       | 93  |
| <b>OMG</b> Object Management Group . . . . .               | 27  |
| <b>OSN</b> Online Social Network . . . . .                 | 30  |
| <b>OSS</b> Open Source Software . . . . .                  | 6   |
| <b>P2P</b> Peer to Peer . . . . .                          | 1   |
| <b>PLE</b> Personal Learning Environment . . . . .         | 130 |
| <b>PoS</b> Proof of Stake . . . . .                        | 17  |
| <b>PoW</b> Proof of Work . . . . .                         | 17  |
| <b>PWA</b> Progressive Web App . . . . .                   | 95  |
| <b>REST</b> Representational State Transfer . . . . .      | 13  |
| <b>RMI</b> Remote Method Invocation . . . . .              | 40  |
| <b>RNN</b> Recurrent Neural Network . . . . .              | 30  |
| <b>ROLE</b> Responsive Open Learning Environment . . . . . | 134 |
| <b>RTE</b> Round-Trip Engineering . . . . .                | 96  |

|  |     |
|--|-----|
| <b>SBF</b> Social Bot Framework . . . . .            | 6   |
| <b>SD</b> Standard Deviation . . . . .               | 47  |
| <b>SLP</b> Service Location Protocol . . . . .       | 13  |
| <b>SNS</b> Social Networking Site . . . . .          | 1   |
| <b>SRL</b> Self-Regulated Learning . . . . .         | 32  |
| <b>SUI</b> Structural UI . . . . .                   | 28  |
| <b>SUIT</b> Structural UI with Tags . . . . .        | 28  |
| <b>SUS</b> System Usability Scale . . . . .          | 117 |
| <b>TEL</b> Technology Enhanced Learning . . . . .    | 4   |
| <b>UI</b> User Interface . . . . .                   | 28  |
| <b>URL</b> Uniform Resource Locator . . . . .        | 12  |
| <b>UWE</b> UML-based Web Engineering . . . . .       | 26  |
| <b>V3C</b> Virtus Virtual VET Center . . . . .       | 7   |
| <b>VET</b> Vocational Educational Training . . . . . | 129 |

|  |     |
|--|-----|
|  | 221 |
| <b>VLE</b> Virtual Learning Environment . . . . .                | 31  |
| <b>WebRTC</b> Web Real-Time Communication . . . . .              | 134 |
| <b>WYSIWYG</b> What You See Is What You Get . . . . .            | 134 |
| <b>xAPI</b> eXperience API . . . . .                             | 31  |
| <b>XML</b> Extensible Markup Language . . . . .                  | 104 |
| <b>XMPP</b> Extensible Messaging and Presence Protocol . . . . . | 134 |
| <b>YAML</b> YAML Ain't Markup Language . . . . .                 | 14  |



# Appendix B

## Own Publications

## Relevant Refereed Publications

- [KLN\*20] Ralf Klamma, Peter de Lange, Alexander Tobias Neumann, Benedikt Hensen, Milos Kravcik, Xia Wang, and Jakub Kuzilek. Scaling Mentoring Support with Distributed Artificial Intelligence. In *Intelligent Tutoring Systems (ITS 2020)*, volume 12149 of *LNCS*, pages 38–44. Springer, 2020.
- [LBNK21] Peter de Lange, Lennart Bengtson, Alexander Tobias Neumann, and Ralf Klamma. Blockchain-based Data Verification and Consent Management for Trusted Learning Analytics Using Mentoring Chatbots. In *Learning Analytics & Knowledge (LAK 2021)*, 2021.
- [LFGK17] Peter de Lange, Tracie Farell-Frey, Bernhard Göschlberger, and Ralf Klamma. Transferring a Question-Based Dialog Framework to a Distributed Architecture. In *Data Driven Approaches in Digital Education (EC-TEL 2017)*, volume 10474 of *LNCS*, pages 549–552. Springer, 2017.

- [LGF\*20] Peter de Lange, Bernhard Göschlberger, Tracie Farrell, Alexander Tobias Neumann, and Ralf Klamma. Decentralized Learning Infrastructures for Community Knowledge Building. *Transactions on Learning Technologies*, 13(3):516 – 529, 2020.
- [LGFK18] Peter de Lange, Bernhard Göschlberger, Tracie Farrell, and Ralf Klamma. A Microservice Infrastructure for Distributed Communities of Practice. In *Lifelong Technology-Enhanced Learning (EC-TEL 2018)*, volume 11082 of *LNCS*, pages 172–186. Springer, 2018.
- [LJK119] Peter de Lange, Tom Janson, and Ralf Klamma. Decentralized Service Registry and Discovery in P2P Networks Using Blockchain Technology. In *Web Engineering (ICWE 2019)*, volume 11496 of *LNCS*, pages 296–311. Springer, 2019.
- [LND\*16] Peter de Lange, Petru Nicolaescu, Michael Derntl, Matthias Jarke, and Ralf Klamma. Community Application Editor: Collaborative Near Real-Time Modeling and Composition of Microservice-based Web Applications. In *Modellierung 2016 Workshop Proceedings*, pages 123–127, 2016.
- [LNK117] Peter de Lange, Petru Nicolaescu, and Ralf Klamma. VIRTUS Virtual VET Centre (V3C): A Learning Platform for Virtual Vocational Education and Training. In *Data Driven Approaches in Digital Education (EC-TEL 2017)*, volume 10474 of *LNCS*, pages 500–503. Springer, 2017.
- [LNKJ17] Peter de Lange, Petru Nicolaescu, Ralf Klamma, and Matthias Jarke. Engineering Web Applications Using Real-Time Collaborative Modeling. In *Collaboration and Technology (CRIWG 2017)*, volume 10391 of *LNCS*, pages 213–228. Springer, 2017.
- [LNKK16] Peter de Lange, Petru Nicolaescu, Ralf Klamma, and István Koren. DevOpsUse for Rapid Training of Agile Practices Within Undergraduate and Startup Communities. In *Adaptive and Adaptable Learning (EC-TEL 2016)*, volume 9891 of *LNCS*, pages 570–574. Springer, 2016.
- [LNNK18] Peter de Lange, Alexander Tobias Neumann, Petru Nicolaescu, and Ralf Klamma. An Integrated Learning Analytics Approach for Vir-



tual Vocational Training Centers. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5(2):32–38, 2018.

- [LNNK20] Peter de Lange, Petru Nicolaescu, Alexander Tobias Neumann, and Ralf Klamma. Integrating Web-Based Collaborative Live Editing and Wireframing into a Model-Driven Web Engineering Process. *Data Science and Engineering*, 5(3):240–260, 2020.
- [LNRK19] Peter de Lange, Petru Nicolaescu, Mario Rosenstengel, and Ralf Klamma. Collaborative Wireframing for Model-Driven Web Engineering. In *Web Information Systems Engineering (WISE 2019)*, volume 11881 of *LNCS*, pages 373–388. Springer, 2019.
- [LNWK18] Peter de Lange, Petru Nicolaescu, Thomas Winkler, and Ralf Klamma. Enhancing Model-Driven Web Engineering with Collaborative Live Coding. In *Modellierung 2018*, 2018.
- [LSK120] Peter de Lange, Michał Słupczyński, and Ralf Klamma. Incentivizing Contribution in Decentralized Community Information Systems. In *Companion Proceedings of the Web Conference 2020 (WWW 2020)*, pages 636–644. ACM, 2020.
- [NAK\*21] Alexander Tobias Neumann, Tamar Arndt, Laura Köbis, Roy Meissner, Anne Martin, Peter de Lange, Norbert Pengel, Ralf Klamma, and Heinz-Werner Wollersheim. Chatbots as a tool to scale mentoring processes: Individually supporting self-study in higher education. *Frontiers in Artificial Intelligence*, 4:64, 2021.
- [NLK119] Alexander Tobias Neumann, Peter de Lange, and Ralf Klamma. Collaborative Creation and Training of Social Bots in Learning Communities. In *Collaboration and Internet Computing (CIC 2019)*, pages 11–19. IEEE, 2019.
- [NLK\*20] Alexander Tobias Neumann, Peter de Lange, Ralf Klamma, Norbert Pengel, and Tamar Arndt. Intelligent Mentoring Bots in Learning Management Systems: Concepts, Realizations and Evaluations. In *Learning Technologies and Systems (ICWL 2020)*, volume 12511 of *LNCS*, pages 3–14. Springer, 2020.

## Relevant Non-Refereed Publications

- [KRLJ16] Ralf Klamma, Dominik Renzel, Peter de Lange, and Holger Janßen. las2peer – A Primer, 2016.
- [Lang19] Peter de Lange. Informationelle Selbstbestimmung durch Peer2Peer Technologien: las2peer. In Albert Geukes, editor, *Konferenzband uni.digital 2019*, pages 62–64. Freie Universität Berlin, 2019.
- [NLKK19] Alexander Neumann, Peter de Lange, Michael Kretschmer, and Ralf Klamma. tech4comp Report zu AP 3.1: Technologiestudie einer Infrastruktur zur verteilten Datenanalyse.

## Other Refereed Publications

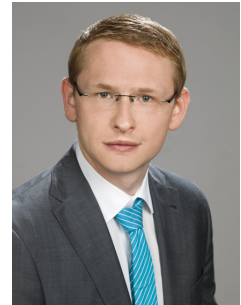
- [LBF\*17] Peter de Lange, Boris Bähre, Christiane Finetti-Imhof, Ralf Klamma, Andreas Koch, and Leif Oppermann. Socio-technical Challenges in the Digital Gap between Building Information Modeling and Industry 4.0. In *Socio-Technical Perspective in IS Development (STPIS 2017)*, CEUR Workshop Proceedings, pages 33–46. CEUR-WS.org, 2017.
- [LNBK18] Peter de Lange, Petru Nicolaescu, Jan Benscheid, and Ralf Klamma. Collaborative Non-linear Storytelling Around 3D Objects. In *Advances in Web-Based Learning (ICWL 2018)*, volume 11007 of *LNCS*, pages 88–98. Springer, 2018.

- [LNBK19] Peter de Lange, Petru Nicolaescu, Jan Benscheid, and Ralf Klamma. Integrating 3D Objects in Collaborative Non-Linear Storytelling on the Web. *Journal of Universal Computer Science*, 25(12):1608–1624, 2019.
- [LSK17] Peter de Lange, Mohsen Shahriari, and Ralf Klamma. Einführung in das wissenschaftliche Arbeiten und Publizieren mittels Blended Learning in der informatischen Fachdidaktik. In *Bildungsräume 2017 (DeLFI 2017)*, pages 385–386. Gesellschaft für Informatik, 2017.



# Appendix C

## Curriculum Vitae



Name: Peter Marcel de Lange

Birthday: January 8, 1988

Birth Place: Eschweiler, Germany

Address: Heisterner Straße 18  
D-52249 Eschweiler, Germany

Email: lange@dbis.rwth-aachen.de

Language Skills: German (native), English (professional), Dutch (fluent)

Professional Experience: PhD Research & Project Work at Chair of Computer Science 5  
(Databases & Information Systems), RWTH Aachen University  
10/2018 - present: *Project worker in BMBF tech4comp*  
12/2015 - 03/2018: *Project worker in EU H2020 WEKIT*  
12/2015 - 12/2016: *Project worker in EU FP7 IP Learning Layers*  
03/2011 - 10/2014: *Student researcher at the Chair for Information  
Systems and Databases, RWTH Aachen University*

Academic Education: 12/2015 - present:  
*PhD Candidate at Faculty of Mathematics, Computer Science and  
Natural Science, RWTH Aachen University*  
10/2013 - 09/2015:  
*M. Sc. Computer Science at RWTH Aachen University*  
10/2008 - 09/2013:  
*B. Sc. Computer Science at RWTH Aachen University*

