DevOpsUse: A Community-Oriented Methodology for Societal Software Engineering

István Koren

Abstract The demanded fast innovation cycles of the ongoing digital transformation create an unstable environment in which the demands of heterogeneous professional communities need to be addressed. Moreover, the information systems infrastructure of these professional communities has a strong influence on their practices. However, the evolution of the web as infrastructure is shaped by an interplay of new technologies and innovative applications. It is characterized by contrasts, such as centralized versus peer-to-peer architectures and a large number of end users versus a small number of developers. Therefore, our aim is to stabilize these dichotomies apparent in the web by means of an agile information systems development methodology. The DevOps approach promotes stronger cooperation between development and operations teams. Our DevOpsUse methodology additionally fosters a stronger involvement of end-user communities in software development by including them in the process of infrastructuring, that is, the appropriation of infrastructure during its usage. The developed DevOpsUse methodology and support tools have been successfully validated by the transitions between three generations of technologies: near real-time peer-to-peer web architectures, edge computing, and the Internet of Things. In particular, we were able to demonstrate our methodology's capabilities through longitudinal studies in several large-scale international digitalization projects. Beyond web information systems, the framework and its open-source tools are applicable in further areas like Industry 4.0. Its broad adaptability testifies that DevOpsUse has the potential to unlock capabilities for sustainable innovation.

I. Koren (✉)
Information Systems and Databases, RWTH Aachen University, Aachen, Germany
e-mail: koren@dbis.rwth-aachen.de

M. Felderer et al. (eds.), *Ernst Denert Award for Software Engineering 2020*,
https://doi.org/10.1007/978-3-030-83128-8_8

1 Introduction

The profound digital transformation of industrial processes is inevitably leading to more software use. The underlying information systems not only need to be initially developed, but they also have to be maintained. Shorter time-to-market processes and far-reaching system integration additionally make it necessary to increase the number of updates. To address this challenge, there have been tremendous advances in software engineering methodologies over the past few decades. While historically the waterfall model has been adopted for the strict process from formal contract to product, it is now being replaced by agile methods. Technology support has also followed this development. Modern frameworks are driving the separation of concerns even further. This has resulted in component-based architectures with microservices on the backend and user interface components on the frontend.

Software development, however, is no longer only object of developers. Instead, it has far-reaching implications into the world of business models and processes, and society in general. Therefore, the question is whether current methodologies can cope with the increased speed and widespread societal involvement. How to incorporate modern aspects such as increased agency of end-user communities and data sovereignty? Especially with regard to the end users, we notice that even in agile methods like Scrum, the users are only at the beginning and at the end, that is, they are largely detached from the actual development. In our research we have thus developed a methodology that explicitly integrates end-user communities. Our solution is characterized by the deep integration of collaboration tools, as well as the application of peer-to-peer architectures. At the same time, contextual forces such as changing technologies need to be stabilized in order to allow a sustainable development process. The evolution of information systems from mainframes to PCs and cloud systems leads us to *societal software*, which increases the responsibility of its users by paying as much attention to the process of creating software as to the software product itself [46]. The implications of the research presented here go beyond the technical aspects and open up new interesting questions that extend into operational and legal perspectives.

This chapter is structured as follows. In the next section, we discuss the motivation behind our research. Section 3 then presents the methodology in detail. Section 4 provides an evaluation of technological and methodological aspects, before discussing implications for societal software development projects. Finally, Sect. 5 concludes this chapter and gives pointers on future work.

2 Motivation

The unrestrained demand for software products together with fast development cycles lead to many challenges. Rapid innovation cycles and changing technology create a disruptive and unstable environment in which the requirements of endless

communities must be met. The shift in speed becomes evident when considering the update rates in the newly established app market economies of mobile operating systems. The number of available developers alone cannot satisfy this demand. Research fields such as *End User Development* attempt to solve this dilemma by putting tools in the hands of users to build software themselves [34]. The shift toward societal software development mentioned in the previous section extends this end-user integration and expands it to the entire methodology.

Information infrastructure plays a special role here. The general term *infrastructure* thereby refers to an underlying factor. Information systems infrastructure, while only partially visible and thus hard to grasp, has a strong influence on user and developer practices. Driven by a body of standards, the web has reached significance not only of technological nature but also quite distinctly of a societal dimension. Its proliferation highlights the ubiquitous nature; it is now available everywhere, on various types of hardware. Constantly evolving standards thereby ensure interoperability between manufacturers and devices. Today, smartwatches have built-in browsers, industrial assets are controlled by web interfaces, and even the touchscreen control panels of the latest generation of space capsules work with web technologies like JavaScript and HTML.[1] Conceptually, the web is a graph of linked resources [40]. Open interfaces allow the composition of these linked resources to form distributed services and apps. However, changing interfaces can also make them drift apart. One of the web's key strengths is therefore also among its weaknesses: the continuous context changes do not only increase the web's applicability and adoption, but also require constant retraining of users, developers, and operators in order to handle the new realms.

Figure 1 highlights current dichotomies in web information systems engineering. On the left, we see the everlasting duality between centralized and distributed technologies,[2] plus the combination of those. On top, device innovations create a constant need for software adaptation, frameworks, and even usability considerations. On the right, the imbalance is portrayed between a small number of developers who know *how* to create software versus a large number of end users who as domain experts know *what* they need. Finally, the bottom layer refers to the ongoing changes in workplace settings caused by digitization. Connecting all of these aspects, the challenge is to create a core that holds and links everything together.

2.1 Central Hypothesis

In this field of mutually influencing dichotomies and the underlying infrastructure, several research questions arise. What are the building blocks of community-

[1] cf. https://cnet.co/3fiK0V5.

[2] The reader is kindly referred to the history of computers from mainframes to personal computers to the cloud, back to current edge computing efforts.

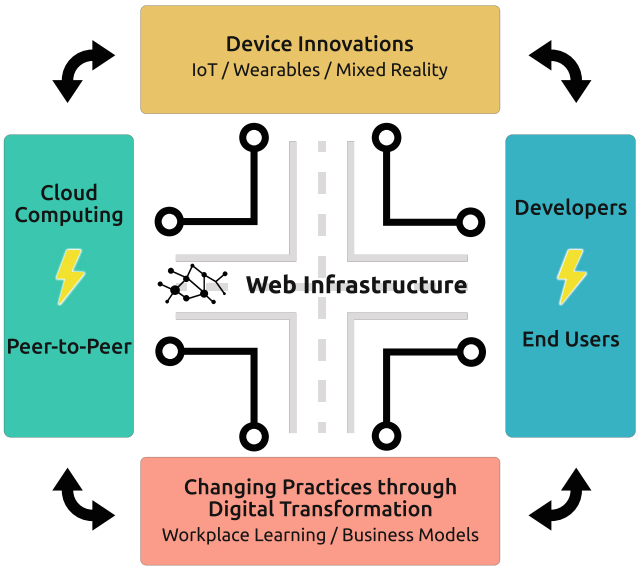

Fig. 1 Dichotomies in web information systems engineering

oriented information systems? How can we enable communities to develop information systems on their specific information systems infrastructure? How to create a sustainable life cycle of the developed information systems? Our central hypothesis is that we can provide a stabilized socio-technical infrastructure on top of the web as open ground. We therefore augment the collaborative notion of DevOps as automation-driven cooperation between developers and operators by the notion of end users. The goal of this extension to DevOpsUse is to make information systems more resilient to technological disruptions by continuously engaging their users. We provide automation of end-user participation via tools for social requirements engineering and service deployment, among many others. Throughout this chapter, we give selected pointers to these tools; for the in-depth discussion and answers to the above questions we refer to the dissertation [26]. As overall methodological research framework, we work along the design-science methodology by Hevner et al. [17]. The seven guidelines tackle the problem-solving process by building and applying an artifact that is later evaluated with due rigor. Instead of a single design artifact, we created multiple particular tools that we then connect in the overarching methodology.

2.2 Research Background

Agile practices in software engineering promote a stronger focus on the social aspects like the development team and the customer. Additionally, the mindset of the

Agile Manifesto acknowledges frequent changes and overall working software [12]. Most popular instantiations include, for instance, *Extreme Programming*, *Scrum*, or *Kanban*. *DevOps*, as a clipped compound of *development* and *operation* teams, is driving a stronger cooperation between these by extensive automation. Recently, related concepts have been introduced, like *DevSecOps* that stresses the growing importance of security. We argue, however, that these methodologies do not explicitly integrate users into the development process itself. In Scrum, for example, users appear at the beginning and end of each sprint. Integrating end users, more specifically *Communities of Practice* (CoP) as groups of professionals working toward a common goal [53], helps not only leveraging their domain knowledge but also increasing their agency and involvement, thereby sustaining the development results.

Approaches that integrate end users are categorized as End User Development (EUD) [34]. According to Liberman et al., there are two possible realizations: parametrization of software products and creation from scratch. The research domain of EUD is rather concerned with the second. Numerous ways have been introduced, like macros for automating tasks in office applications and programming-by-example in smart home settings. Yet they all specifically target an application case and do not extend their findings to the methodological core. In our research, we look at the underlying structures supporting information systems: the infrastructure. Generally spoken, an infrastructure is "an underlying base or foundation especially for an organization or system" [1]. In the obvious analogy of traffic infrastructure, road networks connect cities and countries to ports and other continents. With this, we can exemplify the transitory nature of infrastructure: what is infrastructure for one (driver) is the work item for the other (road worker). This transition is intrinsically much faster in software engineering, where today's developer tools render it possible to build a simple application and scale it to thousands of users in the matter of a few days or hours. In information systems literature, the term *Infrastructuring* has been coined [41, 51] to signify the creation and continuous adaptation process. It represents *in situ* design work, respectively *design-in-use* as opposed to *design-before-use* [42]. *To infrastructure* emphasizes the conditional, flexible and open character of the infrastructure design process [51]. Thereby, the creation process is shaped by conventions of practice. At the same time, the demands of professional communities are under constant change, so their designs are expected to evolve with them [5]. This highlights the need for better collaboration between communities and the developers and operators supporting them. Communication is considered an essential part of infrastructuring that acts as a bridge between actors and resources in different contexts and practices [35]. Similarly, the gap between users and designers is one of the major challenges in design [47].

It is costly to put a lot of work in features that are not needed. *Open innovation* tackles the circumstance that ideas are often planned without meeting the real requirements, by opening up the ideation process to external influence [9]. However, the duality between being too closed and too open may harm the original business model of companies. For a sustainable open innovation strategy, we argue that

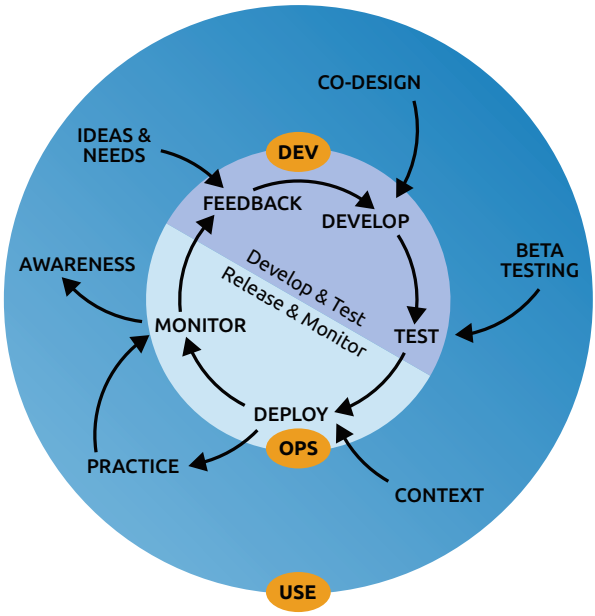

Fig. 2 The DevOpsUse life cycle has the DevOps model at its core, while aspects of end-user communities are surrounding and influencing it

the opening must be well-integrated into the methodological foundation. This is impressively demonstrated by the open-source movement in the software industry, scaling to a massively distributed, open effort. Tuomi therefore sees open innovation to be strongly related to open-source software [52]. In this context, Hippel's *lead users* [18], who stand for domain experts acting as innovators, need to be included.

In this section, we explained the theoretical backgrounds that have significantly influenced our research. In the following, the DevOpsUse methodology is explained in detail.

3 DevOpsUse Methodology

One of the main parameters that influence the velocity in software engineering in the context of frequent changes is the choice of the software development methodology. This area has already seen a great deal of progress in the last decades: The shift from inflexible waterfall models to agile environments has made a significant contribution toward dealing with change. DevOps is a recent concept that furthermore includes operators and thus provides a holistic view on development and deployment [11]. In this approach, automation between these groups plays an important role in resolving the inherent conflict between them. At the core of Fig. 2 we see the simplified life cycle of the cooperation between developers and operators. The arrows represent steps involving automation. Starting from feedback on the top, that is, the requirements engineering phase, the development of software takes place.

The resulting artifacts are then tested. From the testing phase, the software gets delivered and staged to deployment. Finally, its usage is monitored. According to this conception, the points of connection to users become clear: specifically, at the beginning (feedback) and at the end (monitoring). Conversely, this model is characterized by a lack of attention to end users, as they are not explicitly involved in any of the development or operational phases. The software artifacts produced therefore still have a lot of potential to become even more innovative and user-friendly.

As progressive digitization affects more and more parts of our lives, software also plays an increasingly important role therein. The detachment of development practices from societal processes would therefore become even more critical in the future. We therefore propose *DevOpsUse* as a new methodological foundation for societal software engineering. It adds to DevOps the user as a cross-cutting concern across the whole development and operation cycle. As a vehicle to carry out these ideas, our methodology focuses on the underlying information systems infrastructure, upon which development, operation, and usage are happening. The inclusive development process leverages the collective strengths and potential weaknesses of the people involved.

The outer circle of Fig. 2 exemplifies points of user participation in software development. In the following, we highlight these aspects while going around the circle and present the overarching elements that conceptually connect these points. The first aspect is continuous innovation, that is, the influx of new ideas at every development and usage phase. In particular, we showcase the boundary objects connecting the individual tools.

3.1 Continuous Innovation

Requirements engineering (RE) captures the goals of the users and is the basis for all other development activities [43]. However, common issue trackers are not easy enough to be used by end users. Overall, they are very technical and require to specify details, while for users it is often not evident whether it is caused by a backend or frontend bug. Many open-source projects on GitHub additionally require to follow a strict template with developer-specific terms that are hard to understand.

Social requirements engineering aims at collecting requirements in a way that resembles social networks like Facebook and Twitter [45]. It serves both developers and end users; the latter can easily enter new ideas or bug reports and approve existing ones, while the former can start a dialog with the reporting users through comments. Methodologically similar, *CrowdRE* describes automated or semiautomated approaches to integrate a large number of users into RE [14]. However, CrowdRE explicitly discusses pull and push mechanisms as feedback patterns. Contrarily, in DevOpsUse users are part of a community with developers.

Specifically, our Requirements Bazaar[3] web application extends social RE to web scale. It is a continuous innovation tool that allows a social exchange of ideas while making requirements traceable across ideation, conception, and realization phases. Following the design-science methodology, it was developed iteratively with continuous exchange of its users on the very same platform. The web application runs on mobile and desktop browsers. Filtering functionalities allow to filter only the most or least active requirements, for instance. As a conclusion of this section, we deem continuous innovation principles to be important to keep up disruptive capacities of information systems and to create a sustainable long-term development process.

3.2 Collaborative Modeling

Eliciting the mental representation of stakeholders while being as close to the real world as possible is one of the challenging goals of modeling. At the same time, it is a highly social activity. We thus see models as the smallest common denominator between the domains of developers and end users. In our community-driven approach, the mutual engagement of developers and end users within a CoP helps to build better tools. As a starting point, we take a formal description of REST-based APIs available in service repositories on the web. The OpenAPI interface description language (formerly called Swagger) is a well-known format that is widely used for automatic verification and conformance checks [38]. Along with the goals of model-driven software engineering, it allows the type-safe generation of API access layers for frontend applications. We leverage its expressiveness and created a web-based collaboration tool for wiring services together with user interface components. For this reason, we used the *Interaction Flow Modeling Language* (IFML) that is governed by the Object Management Group, the same standardization organization that oversees the *Unified Modeling Language* (UML). It is a visual domain-specific modeling language for creating visual models of user interactions and frontends [7]. The *Direwolf Model Editor* therefore allows the model-driven composition of APIs and UIs [30]. Technically, it translates the data types described in an OpenAPI description into a palette of possible UI elements, like an HTML list for an array or a label for a string. Similarly, object types that are used as input attribute of a service result in an HTML form being generated. Besides the user interface creation, the tool can support other types of collaborative model creation, as we have demonstrated for example with the *iStar 2.0* strategic goal modeling language [30].

The Direwolf Model Editor is a boundary object between end users and developers. It shows how model-driven methods and associated benefits such as generalization and code generation can be leveraged in a community-driven end-

[3] cf. https://requirements-bazaar.org.

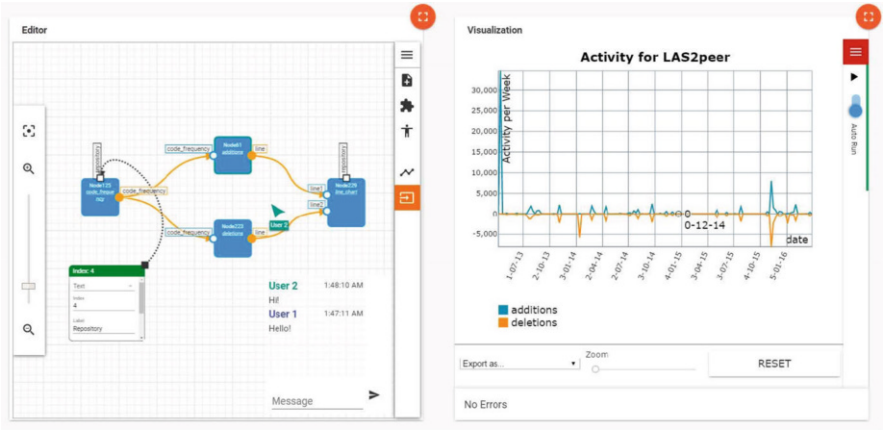

Fig. 3 Screenshot of the SWEVA collaborative visual analytics tool

user development approach. Using generative approaches, repetitive patterns in application creation, like creating input forms based for API inputs, can be scaled to a myriad of users and device types.

3.3 Monitoring

At the intersection of professional communities, the IoT, web services, and peer-to-peer communication between individuals and devices, challenges of analytics are amplified. *Visual Analytics* is the "science of analytical reasoning facilitated by interactive human-machine interfaces" [23]. It facilitates the exploration of large data collections by combining the best of both worlds; computers that can cope with large amounts of data, and humans who can see links and dependencies between two seemingly unrelated datasets.

Since we noticed a lack of open-source or commercial visual analytics tools that are general-purpose, draw on heterogeneous data sources, are community-aware, and can be embedded in community-oriented applications, we developed *Social Web Environment for Visual Analytics* [29]. It visualizes data coming from and flowing between Internet of Things device networks, social (human) networks, and the communication between apps and their components. Its web environment enables a model-driven visual flow design of processing pipelines, which is executed in real time. Thereby, data sources can be anything from real Industry 4.0 machines, body-worn wearable sensors, or input captured on smartphones. Possible methods include, for example, social network analysis like (overlapping) community detection and expert identification.

Figure 3 shows a screenshot of SWEVA. The tool allows different community members to work simultaneously on models and visualizations to support each

other. On the left, the collaborative modeling tool allows to design visual analytics pipelines. The underlying data model is a directed acyclic graph, whose topological ordering ensures that the pipeline can be run without conflicts. It highlights broken nodes to enable quick troubleshooting. The nodes represent either data retrieval operations, custom calculations, or user input. Currently, the tool supports text, number, numerical slider, Boolean toggles, enum selection, and fixed value inputs. On the right, the collaborative visualization tool is responsible for showing the results of the visualization pipeline and influencing its interactive parts by displaying the previously configured user input possibilities. In between these parts sits the core framework, which runs the modeled data processing pipeline. It can either be run locally or remotely on an execution service. All parts or the whole is embeddable into third-party web applications via custom HTML elements. For instance, the `<sweva-visualization-container>` integrates the right part of Fig. 3.

3.4 Connecting the DevOpsUse Life Cycle

In the following, we connect the aspects of the previous subsections with the three particular stages of DevOpsUse, development, operations, and usage. We present aspects of industrial state-of-the-art, related research work and, finally, how we tackled the challenges by showing how they contribute to the overall infrastructure.

Development The role of end users in requirements engineering, either directly through focus groups or indirectly via domain experts, is important by definition. In contrast, end-user involvement during actual programming is much more difficult, as the cognitive hurdle in common textual programming languages is much higher. Increased componentization efforts in software engineering have led to higher maintainability and reduced complexity of individual software packages. This applies to user interfaces just as much as to the encapsulation of many basic app functionalities by libraries. On the backend, microservice architectures similarly lead to higher maintainability and better scalability. To ensure that modularization does not come at the expense of complexity, standards are required for the interfaces. On the traditionally client/server-driven web, the standards can be roughly assigned to the frontend, the backend, and the communication in between. The formal background of standards on the web makes them applicable to model-driven technologies like validation, runtime interpretation, and code generation. User interface components in particular are subject to modeling efforts, as they can be effectively abstracted. Additionally, they are very concrete in terms of the cognitive model of end users, as user interfaces provide the entry point to any application.

Numerous research works have utilized this circumstance by providing formal models as interface description languages. Standardized user interface modeling approaches, for instance, include *Abstract Interaction Objects* [4], *ConcurTaskTrees* [39], and *Cameleon* [8], among others. Prototypes like *Swashup* [36] or frameworks like *ServFace Builder* [37] allow to graphically wire together visual

representations of components. Similar prototypes exist for connecting functionalities in the Internet of Things, for instance, by *RAML for IoT* [24], the open-source *Node-RED* [22], or the commercial *IFTTT* [21].

Similarly, our Direwolf Model Editor allows the wiring of user interface elements with backend functionality. It builds on the idea that developer and end-user communities can support each other. Our tool allows the community-driven creation of frontends by end users with domain knowledge. We are convinced that by enhancing the collaboration between end users and developers, we finally improve the tooling also for developers. In the end, the creation of context-specific, specialized user interfaces, for example, for filling a database in order processes, could be entirely done by the users themselves. Letting domain experts create their own tools puts a focus on their own mental model and understandings. Based on the established formalized descriptions of service APIs, not only graphical but also voice-based user interfaces can be designed, such as for voice assistants.

Operation After a software system is developed, it gets delivered and finally deployed to be executed and used. On a personal computer, software is downloaded and then put on the local hard disk via an installer. For smartphones, the installation process is even easier, as the app is selected in an appstore and with the click of a button, the download and installation happens, after which the app icon appears on the home screen. In web applications users can simply open up a URL and start using the app; with modern *progressive web applications*, the web applications can even be linked on the home screen, with the look and feel of native apps. On the backend, concerning services, users are left out from the possibility of installing apps. The question is therefore, how to allow community members to deploy services on their own, community-specific infrastructure. For example, a learning community in the construction sector could collect photographs of new building material on an in-house server. For this, we leverage containerized microservices. Microservices, first described by Lewis and Fowler and in a blog article [13], combine several advantages. For instance, the decoupling allows them to be developed independently. Defined interfaces (cf. the last section) make sure their compatibility, as they only have to know their deployment URL to connect. Tooling concerning the operation of microservices allows a high degree of automation. Software containers are packages that bundle services together with their libraries, so that they can be run within a sandbox with defined interfaces. This uniform format allows them to be deployed on any host and even makes it possible to change the underlying provider fast. *Docker* containers have reached mainstream adoptions as particular technology that can be run inside clusters in the cloud (e.g., *Kubernetes*).

Thus, to enable end-user communities of practice to deploy their own services, we conceptualized and implemented the *Layers Box*, a host environment for running Docker services. It is a federated cloud-in-a-box that brings industrial-strength container technology in often inexperienced professional communities. One of the main advantages is that CoPs maintain full control over their data, while keeping the authority to decide which data to share. Its high degree of automation allows it to be deployed on different kinds of hardware, that is, local servers, or within

private, public, or hybrid cloud environments. The built-in *Layers Adapter* is a lightweight reverse proxy that accepts incoming service calls over HTTP and forwards them to internally registered services. In the case of a sudden cloud burst, it may also forward requests to previously configured remote Layers Boxes. As additional core part, all Layers Boxes come with a single sign-on solution. For this, we chose the *OpenID Connect* (OIDC) authentication standard, which is built on top of the *OAuth2* authorization framework. OIDC is also supported by a number of online account providers like Google, Auth0, or the German netID. Under the hood, our OIDC server can connect to existing LDAP or Shibboleth user directories.

When deploying services close to professional communities using them, we are entering the field of edge computing [48]. In the literature, typical use cases that leverage the low latency on the edge are analytics [49], machine learning, or visual applications in the area of augmented and virtual reality [15]. Another possibility to reduce latency by offloading applications from the cloud is to use peer-to-peer architectures. Peer-to-peer systems break up the dichotomy of client and server. In use cases where large chunks of data need to be transferred through the network, resources can be saved by directly forwarding data on the shortest topological path. Another advantage is increased privacy, as data does not need to be routed over a central entity that can possibly intercept message content. We developed a number of tools targeted to end-user communities that leverage recent web standards to allow browser-to-browser communication [26]. We were able to show that by providing abstractions in terms of library supported, the increased complexity can be managed well. Besides, advanced standards on the web render service discovery of local Internet of Things devices possible, without the indirection of a cloud. In particular, we connected the ideas of end-user development and the IoT [28].

Usage We conclude the DevOpsUse life cycle by focusing on its usage aspect and, in particular, on analytics functionalities. Gartner reports several commercial tools for analytical reasoning [20], for instance, *RapidMiner* and *Tableau*. *KNIME* is another visual workflow builder for interactive data analytics [3]. There are also tools specialized on visualizing aspects of the Internet of Things, like the *IBM Watson IoT Platform* [19] and the *Bosch IoT Suite* [6]. While DevOps focuses on metrics provided by the host environment that are interesting to developers and operators, DevOpsUse extends the approach to integrate end users by giving them tools for self-monitoring. Through collaboration and awareness functionalities, multifaceted visual analytics with possibly conflicting views about interpretation of results can be carried out. Our SWEVA tool allows to collaboratively design processing pipelines while accessing a variety of community-specific data sources. We thereby leverage visual analytics that combines the power of computer-generated analytics and human interpretation. The approach is universally applicable and easy-to-use and runs on all web platforms, even on constrained devices, as processing can be offloaded to more powerful nodes running microservices. As a use case spanning the mentioned interplay of IoT, human, and services, we demonstrated its usability within the *Immersive Community Learning Analytics* scenario portrayed in Fig. 4. Learning analytics aims to collect, manage, analyze, and exploit data from learners

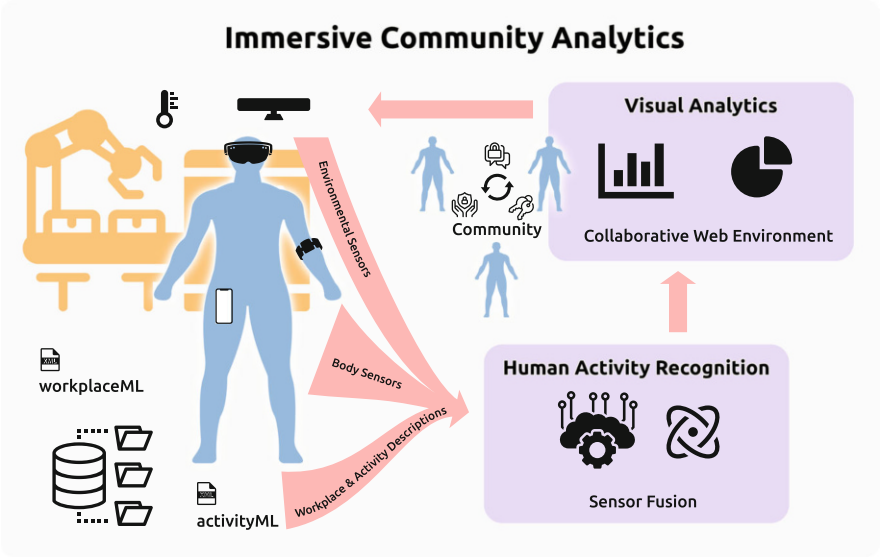

Fig. 4 Immersive community analytics of human activities on the shopfloor

and instructors to facilitate the actual learning process [25]. It connects body-worn sensors described by the ARLEM standard with a data processing infrastructuring running sensor fusion. The results are visualized within the web browser running in the augmented reality headset.

For an in-depth discussion on community-aware analytics capabilities, for example, including the design of community information system success measures, we refer to the dissertation [26]. Generally, our methodology and tool acknowledge the collaboration between involved stakeholders. It is a concern that influences all phases of software engineering. Although it is in the nature of stakeholder collaboration, advanced real-time collaboration capabilities need to be made explicit and integrated into development support tools.

4 Methodological and Technical Evaluation

We evaluate our methodology regarding three aspects. First, we look at three major advancements of the web on a technological level; our framework was not only able to handle but even to support them. We then show how evidences of DevOpsUse tools and processes can be found in a real societal research and development projects and present best practices. Finally, we look at the inherently more complex area of Industry 4.0 and show how DevOpsUse relates to it and provides a path for its continuous innovation.

4.1 Technology Evolution

Starting as a document exchange platform between researchers at CERN, the web has come a long way and is now spreading into more and more areas. The speed of its proliferation can be noticed by the conceiving and implementation of new standards. This frequently changing context makes it hard to build on top of it. With DevOpsUse, we were able to tackle three generations of technology that were integrated into the web's infrastructure over the last decade: peer-to-peer computing, edge computing, and the Internet of Things. We thereby show that we do not only target communities but can also handle technological leaps well. In the following, we shortly discuss each of them.

Near Real-Time Peer-to-Peer Computing The client/server-driven web is generally orthogonal to peer-to-peer technologies, which aim for direct connections between two computing devices. Among consumers, peer-to-peer has long since entered the mainstream, although it was initially tainted due to major file-sharing lawsuits. Today, applications include video conferencing, blockchain technologies, and locally shared folders. On the web, the Web Real-Time Communication (WebRTC) standard made browser-to-browser messages possible around the year 2013 with Google's Chrome browser. In 2017, Apple and Microsoft followed with their own implementations, and only recently, in January 2021, the version 1.0 of the standard was announced by the W3C and IETF organizations. We evaluated the technology early on and were able to cut browser-to-browser roundtrip latency from around 150 ms to around 25 ms in a local network [27]. Specifically, following the methodological core of building upon standards, we were able to replace the connection layer of a collaborative multi-display user interface from a client/server to a peer-to-peer architecture, without touching the user interface source code itself. This enabled new use cases like gaming across browsers.

Edge Computing With the Layers Box, we pioneered self-managed installations of services on-premise. In the meantime, the open-source Kubernetes platform has taken over the market rapidly. Serverless computing is a next evolutionary step in the history of componentization and modularization that microservice architectures pioneered for backend services. They further encapsulation service modules into dedicated functions, each responsible for a single API call. This makes onboarding new developers easier, as no large monolithic technology stack needs to be learned to integrate new functionality. As it is possible to further package these into Docker containers, they can be easily integrated into our Layers Box. The web is currently undergoing another technological transition from resource-oriented to query-based service interfaces. In this evolutionary step, the GraphQL framework for query-based API access gains popularity. In a recent work, we were able to provide automated transformations from the previously used OpenAPI stack to GraphQL [31]. This relatively simple step allows all of our end-user modeling tools to still be used.

Internet of Things The model-based approach of connecting the API description language OpenAPI to IFML as described in Sect. 3.2 can be extended to the Internet of Things as well. For that, we leverage the AsyncAPI documentation convention that describes asynchronous event-based architectures as they are common in the Internet of Things [2]. Again, in our tools, the replacement is a minor step, yet it enables entirely new use cases.

The societal impact of each one of these technological steps is profound. Beyond web information systems, the framework and its open-source tools are applicable in further innovative areas like mixed reality and Industry 4.0.

4.2 Best Practice Guidelines

We were able to demonstrate our methodology's capabilities through longitudinal studies in several large-scale international digitalization projects. Additionally, scalability and involvement aspects were confirmed in entrepreneurial and medical teaching courses. In the former, our student researchers acting as developers (computer scientists) were asked to use and evaluate tools like Requirements Bazaar. In the context of the latter studies, medical students in turn used the end-user-oriented tools. Most of societal research problems require complex information systems that need to be developed. Due to the involvement of our research group in multiple European projects in the area of Technology-Enhanced Learning (TEL), it was obvious to analyze their technology development. The area of lifelong learning in TEL is particularly interesting to aspects of societal software development, as its main subjects are humans and their learning capabilities in changing professional environments. The speed at which new skills are needed as workplaces continue their digital transformation is increasing.

Figure 5 portrays information system design and development activities within the Learning Layers project of the European Commission (Framework Program 7, runtime 2012–2016). They are put into context around the DevOpsUse life cycle. The tool development was partitioned into four co-design teams. The precise team descriptions can be found in a previous publication [33]. Two of them (Bits & Pieces and PANDORA) tackled societal issues in healthcare, while the other two were dealing with the construction sector (CAPTUS and Sharing Turbine). To analyze their processes of information system development, we collected data points like their initial requirements selection. Additionally, we gathered numerous artifacts left behind by the design teams, including various pages created and updated in the project wiki, text documents shared in the collaborative cloud space, as well as photos, videos, and audio recordings distributed within the project. Overall a flow of information (domain knowledge) comes in from the left, while on the right, developed artifacts and material can be seen. For instance, CAPTUS performed a market study. PANDORA worked with interviews of end users and developers. The Sharing Turbine team organized group workshops. Bits & Pieces created user interface mockups and discussed them with researchers.

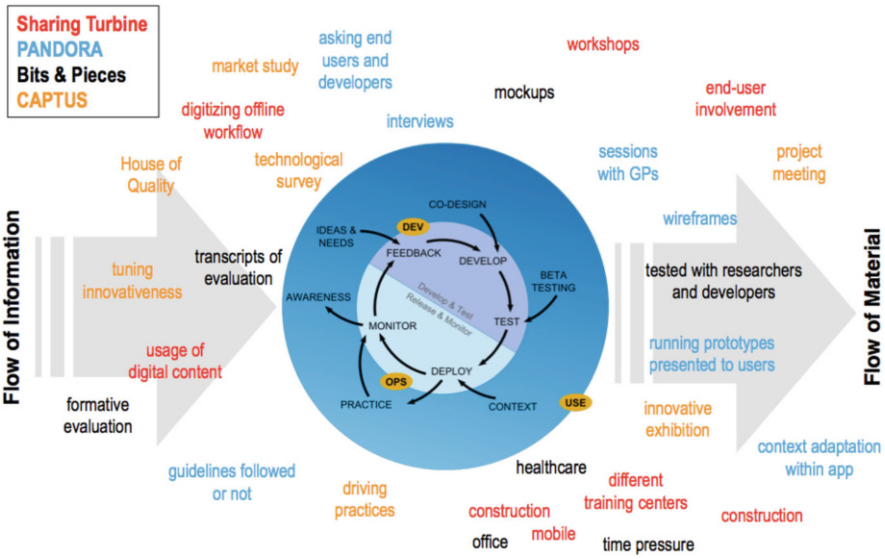

Fig. 5 DevOpsUse case study with four co-design teams

Following our experiences in these and multiple other research projects dealing with societal matters, we distilled best practices and recommendations to tackle common software engineering challenges, which we presented in detail earlier [46]. Here, we shortly outline the main recommendations. Generally, they can be divided into social and technical instruments. Social aspects play a major role in community-driven information systems development. We therefore set up two subcommunities. One is the *developer task force*, a group of developers that regularly meet to tackle everyday issues in software development. The other is a governing body or *architecture board* that decides with wider, often strategic impact on the project.

Concerning the technical setting, we suggest multiple building blocks. The technological development infrastructure needs to be standardized across participating organizations. Following the ideas of open innovation in open-source systems [52] they work best when the pivotal point is institutionalized, that is, central information systems need to be set up and fixed early on. For instance, it includes services for source code management and versioning, continuous integration, continuous delivery, continuous deployment, as well as continuous innovation by tools like Requirements Bazaar. These systems should be interconnected via means of automation, for example, to perform regression tests. The particular software systems need to be decided early on to not hamper the initial development efforts. However, they should not be understood as a fixed entity that cannot change over time and across projects. Recommendations change over time because of the everlasting duality between social and technical development. Pointers to particular software are highly susceptible to changes in the tool environment and licenses. Overall, integration should be a convergent force on three layers. First, social

integration should happen with application partners, end users, and domain experts. Second, server-side integration ensures that services are compatible to each other. Third, client-side integration makes it possible to share data across apps.

As a one-stop shop for interested open-source developers, a developer hub should collect all project documentation and resources like libraries. These dedicated websites collecting all of an enterprise's offers for developers or integrators are also pursued by large companies like Google and Amazon. For instance, in our experience, video tutorials with accompanying textual documentation about particular APIs work well as teaching material. Additionally, in our case, we were able to retarget these videos for teaching DevOpsUse in our entrepreneurial lab course [10].

4.3 Application in Industry 4.0

In the last section we discussed the application of DevOpsUse in the realm of large-scale societal development projects. An area that is even harder to manage is the inherently complex Industry 4.0 setting. The term Industry 4.0 refers to the fourth industrial revolution, driven by digital transformation and characterized by data-driven insights [32]. Figure 6 discusses a typical setting in industrial companies today. The environments are divided into the *development*, *production*, and *user*. While in information systems development there are numerous programming languages, integrated development environments, and runtime frameworks, the production (planning) landscape is characterized by an even more diverse set of design and planning software, file formats, product, and runtime specifications. This leads to disruptive and incompatible data exchange. For instance, data and models are only available within proprietary systems and not ready for cross-domain use.

These incompatibilities in information systems can be generally considered less resource-intensive compared to asset-heavy industrial settings. Still, modern production settings heavily rely on software, making particular aspects of DevOpsUse applicable. In the interdisciplinary cluster of excellence *Internet of Production* at RWTH Aachen University that started in 2019, we are currently actively implementing the methodological findings of our research. Here, data plays a much larger role than in the socio-technical systems of TEL that were discussed in Sect. 4.2. First achievements were setting up a large-scale Kubernetes-based server cluster that is able to instantiate services in Docker containers. Within this cluster, we have put in place multiple databases to create a *data lake* [44] that ingests raw data from production and makes it available for later data-driven operations. Following our own recommendations, a central identity provider (Keycloak) is authenticating and authorizing human users, and later industrial assets that want to push data into the data lake. With the help of model-based technologies we want to automatically generate data schemas from SysML descriptions. Machine learning algorithms will then be able to work on the data to generate (real-time) insights to automate processes.

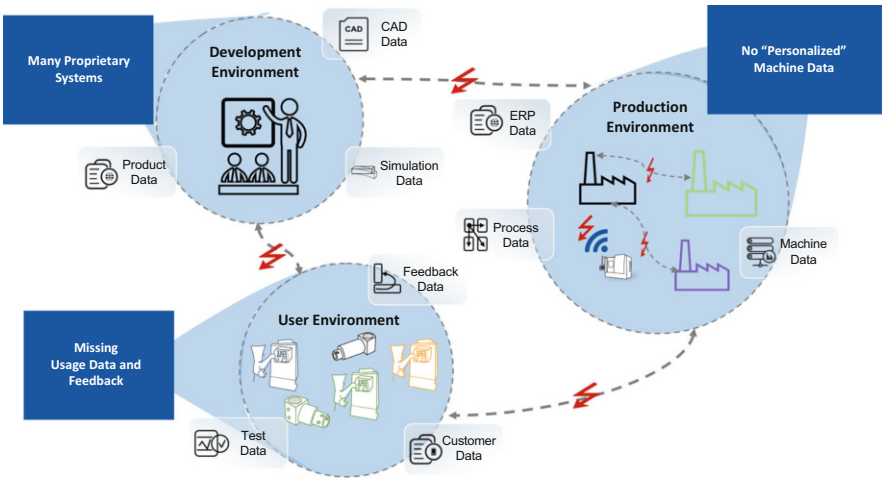

Fig. 6 Challenges of production settings

While this work is still at its infancy, we have already made significant progress setting up open-source web technologies. For instance, we are currently evaluating the use of the new bytecode standard WebAssembly to uniformly target computational use cases on the edge and on the cloud. Another example is the use of GraphQL as the primary access layer to the data lake. Finally, the adoption of further means of the DevOpsUse methodology, like continuous innovation and automation, may unlock the full potential that Industry 4.0 promises in terms of productivity [50].

5 Conclusion

During the dissertation project, the full extent of which we could only touch on here, we developed a methodology and tool support that stabilizes the conflicting aspects evident in the development of information systems. With the advent of societal software, development processes have become much more complex and engineering methods have to consider informal structures of Communities of Practice much more than before. Our community-oriented development life cycle DevOpsUse acknowledges that existing agile methods do not integrate end users to the full extent. Using a digital ethnography approach, where we as researchers took part, we validated our findings in several large-scale societal development projects and their professional communities of practice. Our tools are available and actively enhanced as open-source solutions on GitHub.[4] Lack of interoperability between

[4] cf. https://github.com/rwth-acis.

new and existing tools was tackled by relying on standardized, open interfaces from industrial practice. Each software developed features synchronous remote collaboration capabilities to stress the collaborative nature of infrastructuring within communities [16].

Our work is based on the fundamental insight that communities work on their specific but web-based infrastructure. Therefore, we have been guided by infrastructuring theories from information systems and adjusted parameters on top of it, while pushing established boundaries like in the case of peer-to-peer technologies. The artifacts were created and communicated following the phases of the design science in information systems guidelines [17]. For instance, we presented and discussed results at several summer schools in the area of technology-enhanced learning, as well as the open-source community at venues such as FOSDEM. Additionally, we carried out our research together with numerous students of our technical university, for example, in yearly practice-oriented lab projects, where students work together with local high-tech startups.

We validated DevOpsUse with three technological shifts that happened on the web, namely peer-to-peer technologies, edge computing, and the Internet of Things. At the intersection of these, technical improvements such as reduced latency, economical merits, and even privacy aspects can be considered. Beyond the demonstrated technology-enhanced learning projects, our findings can be applied to other societal and industrial aspects of information systems development, such as Industry 4.0. This opens up several interesting new challenges. We are working on implementing the methodology in industry. Here, the impact of web technologies is still small, but is expected to increase significantly, driven by artificial intelligence methods that leverage data-driven technologies. In future, software engineering will likely play an even stronger role in cross-functional teams, integrating mathematical and engineering disciplines. Yet, innovation as quality characteristic heavily relies on feedback from multiple sources, in particular those of end users. Therefore, leveraging web technologies, the analytical cycle for instance of industrial manufacturers will extend into the usage cycle, that is, when produced artifacts are used by their customers.

We are convinced that our methodology is employable for future societal challenges and technological leaps as well. Information system development is best dealt with in a societal context, explicitly integrating all community members while keeping their agency and strengthening their involvement. In the end, the principles of far-reaching automation and end-user integration will pave the way for a sustainable societal software engineering.

Acknowledgments Thanks are due to all those who supported me while I was writing my PhD thesis. In the context of this chapter, I would like to thank my advisors Prof. Dr. Matthias Jarke and PD Dr. Ralf Klamma as well as the reviewers for providing helpful feedback. The thesis project has received funding from the European Commission's FP7 IP "Learning Layers" under grant agreement no. 318209, from the European Union's Horizon 2020 research and innovation program under grant agreements no. 687669 (WEKIT), and from the European Union's Erasmus Plus program, grant agreement 2017-1-NO01-KA203-034192 (AR-FOR-EU). Funded

by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2023 Internet of Production—390621612.

References

1. American Heritage Dictionary of the English Language. Houghton Mifflin Harcourt (2018)
2. AsyncAPI: AsyncAPI specification 2.0.0 (2019). https://www.asyncapi.com/docs/specifications/2.0.0/
3. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B.: KNIME—The Konstanz information miner. SIGKDD Explor. Newsl. **11**(1), 26 (2009). https://doi.org/10.1145/1656274.1656280
4. Bodart, F., Vanderdonckt, J.: Widget standardisation through abstract interaction objects. In: In Advances in Applied Ergonomics, pp. 300–305. USA Publishing (1996)
5. Bødker, S., Dindler, C., Iversen, O.S.: Tying Knots: participatory infrastructuring at work. Comput. Supported Coop. Work **26**(1–2), 245–273 (2017). https://doi.org/10.1007/s10606-017-9268-y
6. Bosch: Bosch's IoT platform (2017). https://www.bosch-si.com/de/iot-plattform/bosch-iot-suite/homepage-bosch-iot-suite.html
7. Brambilla, M., Fraternali, P.: Interaction flow modeling language: model-driven UI engineering of web and mobile apps with IFML. The MK/OMG Press. Morgan Kaufmann (2014)
8. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for multi-target user interfaces. Interact. Comput. **15**(3), 289–308 (2003). https://doi.org/10.1016/S0953-5438(03)00010-9
9. Chesbrough, H.W.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business School Press, Boston (2003)
10. de Lange, P., Nicolaescu, P., Klamma, R., Koren, I.: DevOpsUse for rapid training of agile practices within undergraduate and startup communities. In: Verbert, K., Sharples, M., Klobučar, T. (eds.) Adaptive and Adaptable Learning. Lecture Notes in Computer Science, vol. 9891, pp. 570–574. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45153-4_65
11. Ebert, C., Gallardo, G., Hernantes, J., Serrano, N.: DevOps. IEEE Softw **33**(3), 94–100 (2016). https://doi.org/10.1109/MS.2016.68
12. Fowler, M., Highsmith, J.: The agile manifesto. Softw. Dev. **9**(8), 28–35 (2001). http://users.jyu.fi/~mieijala/kandimateriaali/Agile-Manifesto.pdf
13. Fowler, M., Lewis, J.: Microservices (2014). http://martinfowler.com/articles/microservices.html
14. Groen, E.C., Doerr, J., Adam, S.: Towards crowd-based requirements engineering a research preview. In: Fricker, S.A., Schneider, K. (eds.) Requirements Engineering: Foundation for Software Quality. Lecture Notes in Computer Science, vol. 9013, pp. 247–253. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16101-3_16
15. Ha, K., Pillai, P., Richter, W., Abe, Y., Satyanarayanan, M.: Just-in-time provisioning for cyber foraging. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, pp. 153–166 (2013). https://doi.org/10.1145/2462456.2464451
16. Hanseth, O., Lundberg, N.: Designing work oriented infrastructures. Comput. Supported Cooper. Work **10**(3–4), 347–372 (2001). https://doi.org/10.1023/A:1012727708439
17. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004). http://dl.acm.org/citation.cfm?id=2017212.2017217
18. Hippel, E.V.: Lead users: a source of novel product concepts. Manag. Sci. **32**(7), 791–805 (1986). https://doi.org/10.1287/mnsc.32.7.791
19. IBM: Watson IoT Platform (2017). https://www.ibm.com/internet-of-things/platform/watson-iot-platform/

20. Idoine, C., Krensky, P., Brethenoux, E., Linden, A.: Magic Quadrant for Data Science and Machine Learning Platforms (2019). https://www.gartner.com/doc/reprints?id=1-65WC0O1&ct=190128&st=sb
21. IFTTT Inc.: IFTTT (2018). https://ifttt.com/
22. JS Foundation: Node-RED (2018). https://nodered.org/
23. Keim, D.A., Andrienko, G., Fekete, J.D., Görg, C., Kohlhammer, J., Melançon, G.: Visual analytics: definition, process, and challenges. In: Kerren, A., Stasko, J., Fekete, J.D. North, C. (eds.) Information Visualization. LNCS, vol. 4950, pp. 154–175. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-70956-5_7
24. Khodadadi, F., Dastjerdi, A.V., Buyya, R.: Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT. In: 2015 International Conference on Recent Advances in Internet of Things (RIoT), pp. 1–6 (2015). https://doi.org/10.1109/RIOT.2015.7104910
25. Klamma, R.: Community learning analytics—challenges and opportunities. In: Wang, J.F., Lau, R.W.H. (eds.) Advances in Web-Based Learning: ICWL 2013. Lecture Notes in Computer Science, vol. 8167, pp. 284–293. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-41175-5_29
26. Koren, I.: DevOpsUse: Community-Driven Continuous Innovation of Web Information Infrastructures. Ph.D. Thesis, RWTH Aachen University (2020). https://doi.org/10.18154/RWTH-2020-06868
27. Koren, I., Bavendiek, J., Klamma, R.: DireWolf goes pack hunting: a peer-to-peer approach for secure low latency widget distribution using WebRTC. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) Web Engineering. LNCS, pp. 507–510. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08245-5_38
28. Koren, I., Klamma, R.: The Direwolf inside you: end user development for heterogeneous web of things appliances. In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) Web Engineering. Lecture Notes in Computer Science, vol. 9671, pp. 484–491. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_35
29. Koren, I., Klamma, R.: Enabling visual community learning analytics with Internet of Things devices. Comput. Hum. Behav. **89**, 385–394 (2018). https://doi.org/10.1016/j.chb.2018.07.036
30. Koren, I., Klamma, R., Jarke, M.: Direwolf model academy: an extensible collaborative modeling framework on the web. In: Michael, J., Bork, D. (eds.) Modellierung 2020 Short, Workshop and Tools & Demo Papers, pp. 213–216 (2020). http://ceur-ws.org/Vol-2542/MOD20-TuD5.pdf
31. Kus, D.A., Koren, I., Klamma, R.: A link generator for increasing the utility of OpenAPI-to-GraphQL translations (2020). https://arxiv.org/abs/2005.08708
32. Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M.: Industrie 4.0. Wirtschaftsinformatik **56**(4), 261–264 (2014). https://doi.org/10.1007/s11576-014-0424-4
33. Ley, T., Cook, J., Dennerlein, S., Kravcik, M., Kunzmann, C., Pata, K., Purma, J., Sandars, J., Santos, P., Schmidt, A., Al-Smadi, M., Trattner, C.: Scaling informal learning at the workplace: a model and four designs from a large-scale design-based research effort. Br. J. Educational Technol. **45**(6), 1036–1048 (2014). https://doi.org/10.1111/bjet.12197
34. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: an emerging paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development. Human-Computer Interaction Series, vol. 9, pp. 1–8. Springer, Dordrecht (2006). https://doi.org/10.1007/1-4020-5386-X_1
35. Marttila, S., Botero, A.: Infrastructuring for cultural commons. Comput. Supported Coop. Work **26**(1–2), 97–133 (2017). https://doi.org/10.1007/s10606-017-9273-1
36. Maximilien, E.M., Wilkinson, H., Desai, N., Tai, S.: A domain-specific language for web APIs and services mashups. In: Krämer, B.J., Lin, K.J., Narasimhan, P. (eds.) Service-Oriented Computing—ICSOC. Lecture Notes in Computer Science, vol. 4749, pp. 13–26. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-74974-5_2.
37. Nestler, T., Feldmann, M., Hübsch, G., Preußner, A., Jugel, U.: The ServFace builder—a WYSIWYG approach for building service-based applications. In: Benatallah, B., Casati, F.,

Kappel, G., Rossi, G. (eds.) Web Engineering. Lecture Notes in Computer Science, vol. 6189, pp. 498–501. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-13911-6_37
38. OpenAPI Initiative: The OpenAPI Specification: Version 3.0.2 (2018). https://www.openapis.org
39. Paterno, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A diagrammatic notation for specifying task models. In: Howard, S., Hammond, J., Lindgaard, G. (eds.) Human-Computer Interaction INTERACT '97, pp. 362–369. Springer, Boston (1997). https://doi.org/10.1007/978-0-387-35175-9_58
40. Pautasso, C., Zimmermann, O.: The web as a software connector: integration resting on linked resources. IEEE Softw. **35**(1), 93–98 (2017). https://doi.org/10.1109/MS.2017.4541049
41. Pipek, V., Syrjänen, A.L.: Infrastructuring as capturing in-situ design. In: 7th Mediterranean Conference on Information Systems (2006)
42. Pipek, V., Wulf, V.: Infrastructuring: Towards an integrated perspective on the design and use of information technology. J. Assoc. Inform. Syst. **10**(5), 447–473 (2009)
43. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer, Heidelberg and New York (2010)
44. Quix, C., Hai, R.: Data lake. In: Sakr, S., Zomaya, A. (eds.) Encyclopedia of Big Data Technologies, pp. 1–8. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-63962-8_7-1
45. Renzel, D., Klamma, R. (eds.): Large-Scale Social Requirements Engineering, vol. 2. IEEE Special Technical Community on Social Networking (IEEE STCSN) (2014)
46. Renzel, D., Koren, I., Klamma, R., Jarke, M.: Preparing research projects for sustainable software engineering in society. In: Proceedings 2017 IEEE/ACM 39th IEEE International Conference on Software Engineering (ICSE) (2017). https://doi.org/10.1109/ICSE-SEIS.2017.4
47. Sanders, E.B.N., Stappers, P.J.: Co-creation and the new landscapes of design. CoDesign **4**(1), 5–18 (2008). https://doi.org/10.1080/15710880701875068
48. Satyanarayanan, M., Bahl, P., Cáceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009). https://doi.org/10.1109/MPRV.2009.82
49. Satyanarayanan, M., Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Hu, W., Amos, B.: Edge analytics in the Internet of Things. IEEE Pervasive Comput. **14**(2), 24–31 (2015). https://doi.org/10.1109/MPRV.2015.32
50. Schuh, G., Potente, T., Wesch-Potente, C., Weber, A.R., Prote, J.P.: Collaboration mechanisms to increase productivity in the context of industrie 4.0. Proc. CIRP **19**, 51–56 (2014). https://doi.org/10.1016/j.procir.2014.05.016
51. Star, S.L., Bowker, G.C.: How to infrastructure. In: Lievrouw, L.A., Livingstone, S. (eds.) Handbook of New Media: Social Shaping and Consequences of ICTs, pp. 151–162. SAGE Publications, London (2002). https://doi.org/10.4135/9781848608245.n12
52. Tuomi, I.: Internet, innovation, and open source: actors in the network. First Monday **6**(1) (2001). https://doi.org/10.5210/fm.v6i1.824. http://firstmonday.org/ojs/index.php/fm/article/view/824/733
53. Wenger, E.: Communities of Practice: Learning, Meaning, and Identity. Learning in Doing. Cambridge University Press, Cambridge (1998)