IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 29, 2019, accepted April 25, 2019, date of publication May 10, 2019, date of current version May 22, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2916142*

# Enabling Sophisticated Lifecycle Support for Mobile Healthcare Data Collection Applications

**JOHANNES SCHOBEL[1], THOMAS PROBST[2], MANFRED REICHERT[1], MARC SCHICKLER[1], AND RÜDIGER PRYSS[1]**

[1]Institute of Databases and Information Systems, Ulm University, Ulm, Germany
[2]Department for Psychotherapy and Biopsychosocial Health, Danube University Krems, Krems an der Donau, Austria

Corresponding author: Johannes Schobel (johannes.schobel@uni-ulm.de)

This work was supported by funds from the program Research Initiatives, Infrastructure, Network and Transfer Platforms in the Framework of the DFG Excellence Initiative—Third Funding Line.

**ABSTRACT** The widespread dissemination of smart mobile devices enables new ways of collecting longitudinal data sets in a multitude of healthcare scenarios. On the one hand, mobile data collection can be accomplished more effectively and quicker compared with validated paper-based instruments. On the other hand, it can increase the data quality significantly and enable data collection in scenarios not covered by existing approaches so far. Previous attempts to utilize smart mobile devices for collecting data in these scenarios, however, often struggle with high costs for developing and maintaining mobile applications, which need to run on a multitude of mobile operating systems. Therefore, in the QuestionSys project, we are developing a generic (i.e., platform-independent) framework for enabling mobile data collection and sensor data integration in healthcare scenarios. The latter, in turn, is addressed by a model-driven approach, which is shown this paper along with the core components of the QuestionSys framework. In particular, it is shown how healthcare experts are empowered to create mobile data collection and sensing applications on their own and with reasonable efforts.

**INDEX TERMS** End-user programming, mobile data collection, model-driven development.

## I. INTRODUCTION

In a variety of healthcare scenarios, the controlled collection of longitudinal data sets meeting high quality standards is of paramount importance. In this context, well designed *instruments* (e.g., self-report questionnaires) are widely used for collecting data in healthcare studies [1]. Despite well-known drawbacks, data is still collected in a *paper-based* fashion. With the increasing dissemination of smart mobile devices, however, healthcare experts crave for an electronic data collection based on specifically tailored mobile applications. [2] estimates that approximately 50-60% of the overall data collection costs could be saved when relying on electronic instruments instead of paper-based ones, especially regarding long-running clinical trials. Note that studies have already proven that the use of electronic versions of these instruments does not affect psychometric properties [3]. However, it significantly increases the quality of collected data, while at the same time decreasing the time required for data collection [4], [5]. Recent studies further indicate that the use of smart mobile devices in gathering and sensing data might pave the way for completely new findings and insights in medical science [6].

Developing mobile applications for collecting data in healthcare scenarios, however, is a complex endeavor. The mobile application needs to be provided for a broad spectrum of mobile operating systems. Cross-development frameworks can be used to increase code reusability on one hand, but may limit the provided functionality of the resulting mobile application on the other. Furthermore, mobile application developers need to cope with the short release cycles of mobile platforms, resulting in costly code adaptations of already deployed applications as well as the need to support various versions of the same mobile operating system at the same time. In addition, platform-specific peculiarities need to be properly considered to meet user requirements and to obey mobile platform guidelines as well as internal and external sensors need to be integrated in many scenarios. Finally, transferring the logic that guides (untrained) users through the *process of data collection* (e.g., to skip questions

The associate editor coordinating the review of this manuscript and approving it for publication was Alessio Vecchio.

based on given answers or to validate data) from an existing paper-based instrument to a mobile data collection application requires significant communication efforts between healthcare experts and mobile application developers.

In order to deal with these issues, we realized QuestionSys–a mobile data collection framework that offers novel features compared to the state of the art. QuestionSys enables healthcare experts to develop mobile applications for collecting and sensing data from subjects, i.e., they can develop mobile applications without need to involve programmers. For this purpose, a user-friendly *high-level modeling language* was developed, which allows experts to define the logic, layout, and components (i.e., questions) of their instruments on an abstract level. This approach particularly fosters the model-driven development of robust and flexible mobile data collection applications. Based on this, a created model-based instrument may be executed on a variety of smart mobile devices. As this necessitates an advanced *kernel* enabling the model-driven, robust execution of data collection applications, the QuestionSys kernel persists the collected data. Further, it allows extending the functionality of mobile data collection applications by its ability to integrate both internal and external sensors into data collection processes. Moreover, collected data can be retrieved in a well-defined format, relieving experts from manual tasks like digitizing the data collected with paper-based questionnaires to spreadsheets. As another peculiarity, *adaptations* of already running instruments as well as their redeployment to smart mobile devices is provided by QuestionSys. Consequently, healthcare experts are relieved from technical issues related to the proper installation or upgrade process of data collection applications on heterogeneous mobile platforms. The following publications already exist on the QuestionSys framework (end-user programming [7], sensor integration [8], requirements and implementation details [9], [10], and usability studies [11], [12]), but the contributions presented here have not been addressed by these previous works. These new contributions are as follows:

- C1 Detailed insights into the model-driven design principles of the QuestionSys framework are discussed.
- C2 By applying a model-driven approach, healthcare experts shall be relieved from mentally challenging tasks. In order to evaluate whether this can be achieved, an analysis on the perceived complexity when using the model-driven QuestionSys configurator was conducted.

The remainder of this paper is structured as follows: Related work is discussed in Section II. Section III gives insights into the model-driven preliminaries driving the design of the QuestionSys framework, whereas Section IV discusses the applied model-driven development phases for mobile applications in healthcare scenarios. Section V describes the realized technical components of the QuestionSys framework, whereas Section VI deals with aspects along the model-driven development phases. Results from the analysis on the perceived complexity are presented in Section VII, while Section VIII concludes the paper with a summary and an outlook.

## II. RELATED WORK

In the context of the present paper, three categories of related work are particularly relevant. First, we need to discuss model-driven approaches focusing on the development of mobile applications. Second, we need to relate our work to general approaches dealing with the development of mobile applications for collecting and sensing data. Third, we discuss related mHealth applications.

### A. MODEL-DRIVEN APPROACHES FOR DEVELOPING MOBILE APPLICATIONS

Model-driven development has raised interest in research and practice since its beginning [13]–[15]. Our work is related to model-driven approaches for developing mobile applications in general and mobile data collection applications in healthcare. While model-driven development has been a well-established principle for desktop and server applications for more than a decade, only few approaches applying it to smart mobile device applications exist.

An approach being noteworthy is *Google App Inventor* [16], which enables an abstract view on mobile application development. More precisely, its editor provides colored blocks representing different code fragments, which may be graphically composed to *develop* a more complex mobile application. In turn, the blocks are then transformed to native Android code (i.e., Java code fragments), which is then executed on Android devices. Interestingly, Google stopped the development of this tool, which demonstrated that the high-level configuration of mobile applications constitutes a challenging endeavor, even when only facing one mobile platform.

Other approaches enabling a model-driven development of mobile applications rely on UML diagrams or specific UML profiles. In [17], for example, a UML-based framework for defining mobile applications in a platform-independent way is presented. The framework comprises a model editor as well as an user interface generator. Furthermore, from the created models, program code for the respective mobile application can be automatically generated. As opposed to QuestionSys, this approach aims at relieving IT experts from mobile application programming, but does not intend to involve healthcare experts in the development process. Furthermore, it does not specifically focus on mobile data collection applications in healthcare scenarios.

In turn, [18] presents a WYSIWYG editor for developing mobile applications, which is similar to Apple's Storyboard technique. The underlying model of this approach does not rely on UML, but on a *domain-specific language* for expressing models, which then may be compiled into native language code. Again, application developers shall be relieved from complex programming tasks. A model-driven approach for developing platform-independent mobile applications is presented in [19]: developers describe their

application scenario, entities and device features using a meta-programming language, the resulting models are then translated into the respective native code. Based on the generated entities, backend code for a server component offering common `CRUD` operations is then generated. Most of the configuration, however, is accomplished in a textual way, neglecting the advantages of graphical notations and making its usage difficult for non-programmers. Similar to this approach, [20] provides another textual *domain-specific language* for developing mobile applications based on existing backend web services.

With *XMob*, another *domain-specific language* for the cross-platform development of sophisticated mobile applications is presented in [21]. XMob considers similarities between available mobile platforms and offers a high-level dialect for describing mobile applications. More specifically, it provides components for the *data* and *user interface* layers of respective applications as well as the *events* inter-linking them. Overall, XMob allows creating platform-independent models, which may then be enriched with UML diagrams and be transformed into native code. As a drawback, XMob does not involve healthcare experts in the creation of mobile applications.

An approach accomplishing the latter is presented in [22]. It enables medical staff to model care plans for chronically ill patients. Respective plans are then transformed into a DHTML application, which may then be deployed to smart mobile devices. More precisely, the approach enables physicians without any programming skills to realize specifically tailored mobile applications for their patients with a focus on patient reminders and immediate feedback functions on the patient's status. As opposed to QuestionSys, this approach is domain-specific, i.e., its usage is limited to care plans. An approach for describing services running on smart mobile devices, which is based on XForms, is presented in [23]. It comprises a graphical editor implementing a *domain-specific language*. Any model created with this editor and language, respectively, can then be transformed into a graphical user interface. In particular, the graphical editor allows adapting the generated user-interface as well as the corresponding mobile application during run time as well. Overall, [23] aims at the model-driven development of sophisticated mobile services, which, in turn, may be connected to a powerful backend. Again, the approach does not involve healthcare experts or end users in the procedure of creating mobile applications. *WebRatio* [24], *Mendix* [25], or *OutSystems* are other commercial model-driven platforms that allow users with little programming experience create their models and deploy them to web-platforms or even smart mobile devices based on common web technologies. However, they do not specifically focus on healthcare scenarios.

### B. MOBILE DATA COLLECTION FRAMEWORKS

In general, there exist comprehensive surveys that have identified and elaborated that mobile data collection frameworks are important in the context of healthcare scenarios [26]–[28].

Furthermore, very recent works deal with limitations of smartphones when collecting data in healthcare scenarios [29], [30]. Thereby, several challenging areas have been identified, which may be subject to further investigations.

There exist also works on the quality of the collected data when applying smart mobile technology [31], [32]. However, the shown approaches do not propose a technical solution that can be compared to QuestionSys.

### C. MHEALTH AND SENSING APPLICATIONS

In [33], the benefits of smart mobile devices for collecting data in medical scenarios are discussed. Furthermore, several *mHealth* applications are introduced. In turn, [34] presents a framework that allows collecting data specifically in the context of mental diseases. The platform strongly focuses on the customization of questionnaires (e.g., interval-based questionnaires) and the integration of hardware sensors to increase the data value obtained by the questionnaires. The developed mobile application itself, however, is hard-coded and needs to be adapted manually to emerging requirements or new questionnaires.

An iPad application that enables medical staff to review the health records of their patients during ward rounds is presented in [35]: staff members can dynamically append notes to a record or request additional information. Again, the mobile application is hard-coded and it is restricted to the iOS mobile platform. Reference [36] presents a smart mobile application to capture deviations from standardized care processes. More specifically, medical staff enters data related to the treatment of their patients (e.g., the performed examinations). In turn, this data is then analyzed in order to provide valuable feedback on the treatment of the patient. References [37], [38] combine *WordPress*, a blogging software, and *iBuildApp*, a Web-based application builder, to create a platform supporting students from clinical psychiatry. Although it is possible to provide short questionnaires (e.g., in order to track the students' progress) the main focus of this platform is put on the information retrieval (e.g., psychiatric guidelines).

The web-based configuration platform Sensr for creating simple questionnaires is presented in [39]. As a drawback, the configurator provides only very few elements. Furthermore, the mobile application, used for collecting data, relies on common web-technologies, making it difficult to integrate external sensors.

Finally, there exist open source platforms that support data management in a clinical context. In particular, these solutions often deal with electronic case report forms (e.g., [40]). Although these solutions make use of custom forms, they do not provide flexible support of questionnaires running on smart mobile devices. When considering chronic diseases in a broader context, then various approaches exist that deal with a particular disease. However, they do not address a more generic technical solution [41]–[43]

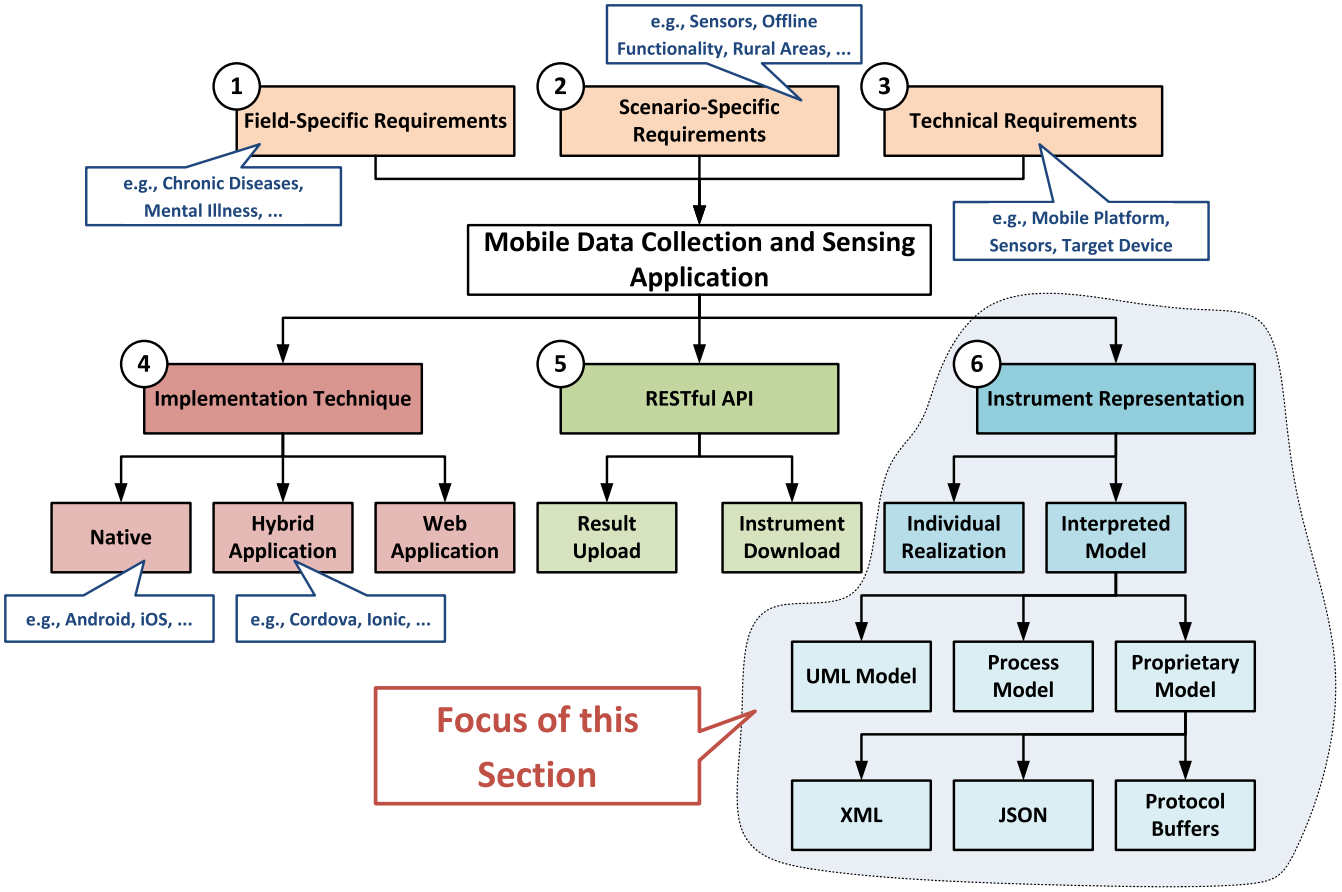



**FIGURE 1.** QuestionSys design criteria.

## III. PRELIMINARIES

QuestionSys applies a model-driven approach for supporting the lifecycle of mobile data collection and mobile sensing applications, i.e., for specifying, configuring, deploying, and executing mobile applications. When designing the QuestionSys framework, several technical alternatives were considered and evaluated along well-defined criteria. The latter had been identified in previous mobile application projects. We categorize them into **six** groups (see Fig. 1).

When developing mobile healthcare applications for collecting and sensing data, not only technical requirements need to be elicited, but also the ones of the specific scenario to be supported. In general, three categories of requirements need to be distinguished (see Categories ①–③): field-specific requirements, scenario-specific requirements, and technical requirements.

Category ① reflects the fact that requirements vary between different healthcare settings, i.e., field-specific requirements must be met. Category ②, in turn, refers to scenario-specific requirements. For example, if data shall be collected in rural areas without a reliable Internet connection, mobile applications should run in offline mode as well. Finally, technical requirements need to be properly addressed (see Category ③). For certain projects, for example, it might be sufficient to implement a mobile data collection application for a particular mobile platform, whereas in other projects multiple mobile platforms need to be supported. We omit a detailed discussion of the three categories and refer to [7]. Three other categories of design criteria are relevant in the context of our work (i.e., Categories ④–⑥ in Fig. 1).

- Category ④: When realizing a lifecycle support framework for mobile healthcare applications, one must choose an appropriate implementation strategy.[1] For the development of the QuestionSys framework, we chose a native implementation strategy, which allowed us to properly address the requirements of the aforementioned categories.
- Category ⑤: For implementing mobile data collection and mobile sensing applications based on the QuestionSys framework, the latter must provide a powerful application programming interface (API). Note that this is crucial to meet the requirements from Categories ① and ②. In the context of QuestionSys, we implemented a RESTful API [45], [46]. Also other works show, that when collecting and managing data in an ubiquitous environment, the provision of a powerful

[1] see [8], [44] for a detailed discussion

API that incorporates the RESTful architectural style is indispensable [47], [48]

- Category ⑥: For any framework supporting the lifecycle of mobile data collection and mobile sensing applications, an appropriate representation of the instruments and their logic needs to be provided. Moreover, the chosen representation format must allow coping with adaptation issues when updating the questions or the logic of an instrument.

While Categories ④ and ⑤ are clear, Category ⑥ needs to be discussed in more detail as it provides the fundamental basis of the model-driven approach applied by the QuestionSys framework. Two questions had to be answered in this context:

1) Shall an individualized application be implemented for each data collection instrument or shall a container application be provided that allows for the (run time) interpretation of the instruments based on their respective representation?
2) Which data structure shall be used to represent instruments in a generic and flexible way

Regarding the first question, we made use of the comprehensive expertises from **eight** projects [7], in which we implemented *individualized* mobile data collection applications. In particular, numerous discussions between mobile application developers and healthcare experts were required to finally meet the field- and scenario-specific requirements. In this context, the changes of the mobile applications were triggered by healthcare professionals, whereas IT experts were needed to implement them, which often led to the well-known business IT alignment gap. To close this gap in the design, implementation and change of mobile data collection and mobile sensing applications, we introduced a novel representation model for instruments, which is understandable to healthcare as well as IT experts, and enables a model-driven lifecycle support for mobile data collection and mobile sensing applications.

Moreover, our aim was to empower healthcare experts to directly work with the representation model of an instrument, i.e., they shall be empowered to understand, create, update, and delete an instrument themselves. In this context, it had to be ensured that healthcare experts can specify the models of instruments covering different scenarios with similar mental efforts. The described empowerment of healthcare experts leads to the second question mentioned above, i.e., to define a representation model for instruments that does not require uncomfortable mental efforts of healthcare experts when creating an instrument. Furthermore, this model must contribute to address the issues identified in the context of the other criteria categories relevant for mobile data collection and mobile data sensing applications (i.e., Categories ①–⑤).

Concerning the choice of the representation model for instruments, Fig. 1 illustrates important design choices: The *UML notation* provides features for graphically modeling classes as well as their semantic relationships. However, UML diagrams are limited when it comes to the *controlled execution* of the corresponding models. *Process models*, in turn, usually come with an easy-to-understand graphical notation as well as a formally specified operational semantics (i.e., formal rules for properly executing instances of the process model). Moreover, a powerful run time environment (e.g., a process engine) is needed, which is able to interpret process models and to execute related instances accordingly. Furthermore, process modeling languages like BPMN 2.0 allow covering different perspectives (e.g., control flow, data flow, resources, and temporal constraints), which are crucial in the context of enterprise information system, but are not needed for properly supporting mobile healthcare data collection scenarios. In the latter context, a more lightweight process model would be sufficient. Obviously, one could also rely on *proprietary models*, represented in various formats (e.g., `XML`, `JSON`, or `ProtoBuf`). However, such solution must always be tailored to a specific use case and, hence, is difficult to maintain, often lacking a clear specification or documentation. Moreover, most likely, some kind of *engine* is needed, which is capable of executing this model. As there is no clear specification, custom adaptations might again cause high implementation costs.

Taking these considerations into account, the various approaches for representing an instrument in a respective model were carefully considered and evaluated.

**TABLE 1. Comparison of model approaches.**

| | Performance | Development Time | Development Costs | Adaptation Time | Adaptation Costs | Execution Semantics | Instrument Logic | Custom Requirements | Sensor Integration |
|---|---|---|---|---|---|---|---|---|---|
| Individual Realization | ● | ◑ | ○ | ◑ | ○ | ◑ | ○ | ● | ◑ |
| UML Model | ◑ | ● | ◑ | ◑ | ● | ○ | ◑ | ○ | ○ |
| Process Model | ◑ | ○ | ○ | ● | ● | ● | ● | ◑ | ● |
| Proprietary Model | ◑ | ◑ | ○ | ◑ | ◑ | ○ | ◑ | ● | ○ |

Table 1 illustrates the different approaches in respect to the relevant criteria [8]. When considering the latter, finally, the process model-driven approach was chosen. However, the selected approach has also its disadvantages. In particular, when considering the fact that a complex process engine (i.e., the container application that interprets instruments) needs to be implemented, resulting implementation efforts must be carefully considered. Furthermore, additional complexity is caused by the need for a modeling tool that allows for the creation of such models. However, the benefits of flexible adaptations and a well-defined execution semantic surpass these disadvantages. Moreover, an extensive and large-scale study with healthcare experts revealed that they were able to understand such a modeling application in a rather short time and on a convenient level with respect to the mental effort (see Section VII).

To conclude to contribution C1 (see Section I), this section provided insights into the model-driven design principles of the QuestionSys framework.

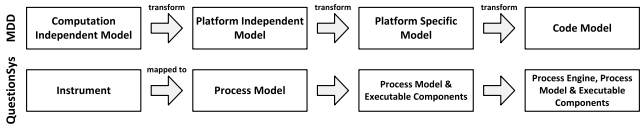



**FIGURE 2. Comparison between model-driven development and the questionsys approach.**

## IV. MODEL-DRIVEN DEVELOPMENT IN QUESTIONSYS

This section discusses the model-driven approach of the QuestionSys framework. In general, a model-driven development follows **four** fundamental phases (see Fig. 2). First, the *Computation Independent Model* may be defined with *Use Case Diagrams*, (functional) requirements, or *Task Trees*. Second, these models may then be transformed to *Platform Independent Models*, like *Sequence Diagrams*, object models, or *Entity-Relationship Diagrams*. Third, in order to specifically address a target platform, this independent model may then be transformed to a *Platform Specific Model*. For example, an *ER-Diagram* is transformed to a *Relational Database Structure*. Finally, a *Code Model* may be generated from the platform specific models. For example, code for `CRUD`[2] operations can be automatically generated [49]. Usually, these transformations (see Fig. 2) require more than one cycle to finally get the appropriate model for a phase. This approach of constantly refining models until code fragments can be derived is one major aspect of a *model-driven development* (MDD). Typically, the *Unified Modeling Language* (UML) is used as a vendor-neutral standard of specifying respective models [50]. As discussed in the previous section, QuestionSys uses process models instead of UML.

To practically identify an appropriate process model, the general prerequisite of any model-driven development was also addressed by QuestionSys: An understanding of the problem to be solved in the given context (i.e., *problem space*) must be created. Commonly, interviews with healthcare experts and/or the realization of real-life projects are the main approaches to create this understanding. Regarding QuestionSys, **eight** healthcare projects have been realized [7] for this purpose. Based on the projects results, a model that describes instruments in a more abstract way and that follows the *model-driven development* (MDD) has been developed (see Fig. 2, *Process Model*). More specifically, an easy-to-understand and high-level abstraction of existing process models was developed. The resulting model is a combination of the ADEPT notation [51], the BPMN notation [52], and newly added aspects based on the experiences from the conducted studies [7]. Following this, the required aspects from the business (e.g., navigation logic) and the technical (e.g., data flow) points of view have been considered for the model (see Fig. 2, *Process Model & Executable Components*) as well. Finally, an automatic transformation was realized that is able to create code fragments automatically [53], which is based on the model created by the healthcare experts (see Fig. 2, *Process Engine, Process Model & Executable Components*).

Technically, QuestionSys implements the four *Model-Driven Development* phases shown in Fig. 2 as follows: In terms of a model-driven approach, the *Data Collection Instrument* serves as the computation independent model. The instrument is then mapped to an executable *Process Model* that can be executed on various platforms (i.e., it serves as platform independent model). The *Process Model*, in combination with so-called *Executable Components* [9], [54], serves as platform-specific model, as they may contain platform-specific features (e.g., accessing sensors connected to a particular smart mobile device type). Finally, the *Process Engine*, together with the *Executable Components* and the *Process Model*, is denoted as the *Code Model*. Note that the process engine is capable of (1) executing the respective data collection instruments (i.e., a process model) and is being able to (2) dynamically load and call the required executable components.

To also conclude to contribution C1 (see Section I), this section has shown in what way QuestionSys adheres to common achievements in model-driven development.

## V. THE QUESTIONSYS FRAMEWORK

The QuestionSys framework follows a model-driven approach. This approach, in turn, allows describing the logic of an instrument (see Fig. 3, ①) in terms of a *process model* (see Fig. 3, ②) that can be interpreted and executed by a lightweight process engine running on smart mobile devices (see Fig. 3, ③) [9], [54]. By applying this approach, instrument logic and application code are strictly separated [55]. The process model acts as schema for creating and executing *process instances* (i.e., instrument instances). The process model itself consists of process activities as well as the control and data flow between them. Gateways (e.g., `XORsplit`) can be used to describe more complex logic within an instrument. Following this model-driven approach, both the content and the logic of paper-based instruments can be mapped to an executable process model. *Pages* of an instrument, thereby, directly correspond to *process activities*; the *flow* between the latter matches the *navigation logic* of the instruments. *Questions* are mapped to *process data elements*, which are connected to activities via `READ` or `WRITE data edges`. These data elements, in turn, are used to store answers when executing the instrument on smart mobile devices. Altogether, the QuestionSys framework applies fundamental BPM principles in a broader context on one hand, thus enabling novel perspectives for process-related technologies. On the other, the executable process model is created through a model-driven development approach.

### A. ARCHITECTURE

In this section, the architecture of the QuestionSys framework is described in more detail.

[2] `Create`, `Read`, `Update`, `Delete` methods to access/manipulate data

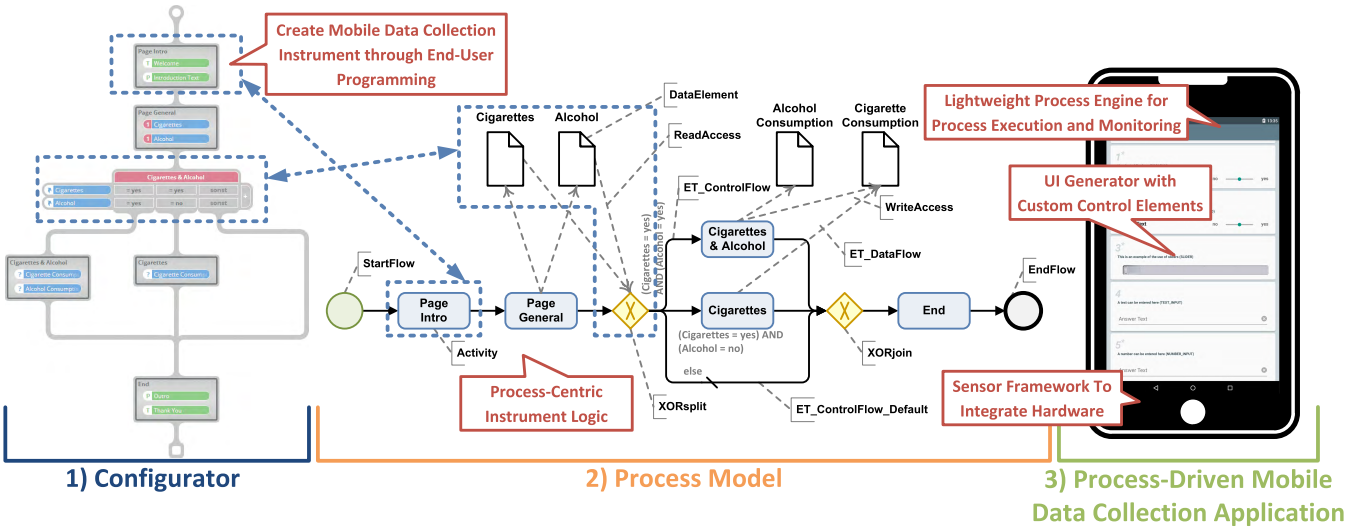



**FIGURE 3.** The questionSys qpproach: (1) modeling a data collection instrument; (2) mapping it to an executable process model; (3) executing it on a smart mobile Device.

#### 1) CREATE DATA COLLECTION INSTRUMENTS USING PROCESS TECHNOLOGY

Data collection instruments are modeled by healthcare experts using a process-aware configurator [10]. This component, in turn, provides the aforementioned and easy-to-understand graphical modeling notation (see Section IV) for healthcare experts to specify the logic of the mobile data collection instrument. Navigation operations influencing the further course of the instrument, as well as the data elements of instruments, can be modeled as well. Data elements, however, are automatically connected to pages, which are important for rendering instruments as they represent single screens on the smart mobile device and allow thematically structuring a questionnaire. In the context of questionnaire instruments, data elements represent questions, whereas navigation operations allows skipping questions depending on previously given answers. Finally, the configurator component allows defining rules for the automated evaluation of gathered data.

#### 2) RELIEVE IT EXPERTS THROUGH AUTOMATIC PROCESS MANAGEMENT

The process model as well as the evaluation rules are mapped to XML documents. The latter are automatically deployed to available smart mobile devices. Collected data, as well as execution information are stored using an XML structure to allow for a subsequent evaluation. Security and privacy is ensured based on state-of-the-art data encryption techniques. The entire communication relies on Web Services [56], [57]. Based on this automation, many challenging requirements of mobile data collection application projects are mitigated. When releasing a new version of an already existing instrument, IT experts are no longer required. Note that release management constitutes the main cost driver in the context of the discussed mobile data collection projects.

#### 3) GENERATE MOBILE APPLICATIONS BASED ON PROCESS MODELS

The process model of a created data collection instrument acts as schema for the execution on the various mobile operating systems. However, this requires the implementation of a lightweight mobile process engine. By interpreting process models directly on smart mobile devices, changes to instruments can be realized in an easy and cost-efficient manner. Note that this provides the basis for flexible adaptations of mobile data collection applications. Finally, instruments are rendered locally on the smart mobile device. The rendering algorithm takes different mobile operating systems, screen sizes as well as languages into account, again utilizing information from the process model.

## VI. MODEL-DRIVEN MOBILE DATA COLLECTION

When realizing mobile data collection applications using the QuestionSys approach, the model-driven development idea is covered along the lifecycle. This section describes techniques allowing healthcare experts to flexibly *develop*, *deploy*, and *execute* data collection instruments on smart mobile devices.

### A. DEVELOPING A COMPUTATION INDEPENDENT MODEL

The configurator component we developed (see Fig. 4) applies process management technologies in a broader scope [58] as well as techniques known from model-driven development to empower healthcare experts to create flexible data collection instruments on their own. This paper only sketches the most important aspects of the configurator component, more details can be found in [10]:

**Element and Page Repository View** (see Fig. 4, left). The element repository allows creating basic elements of a questionnaire (e.g., headlines and questions, see Table 2). The rightmost view shows the editor, where particular attributes of

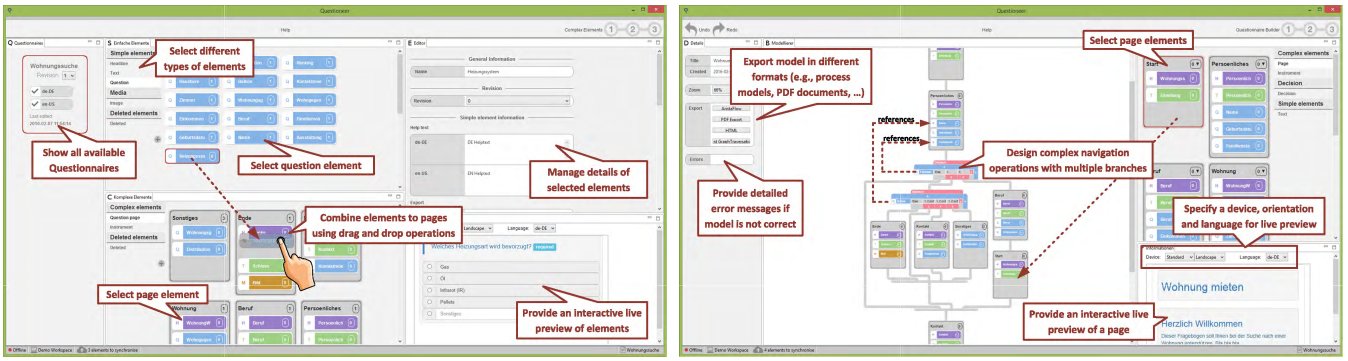



**FIGURE 4.** The questionSys configurator: (left) combining elements to Pages; (right) modeling a data collection instrument.

**TABLE 2.** Element types available in the configurator component.

| | |
|---|---|
| **Data Collection Explanation Elements** | |
| 1. Headline | Introduces the following elements. |
| 2. Paragraph | Provides further details to guide subjects through the data collection process. |
| 3. Media | Provides additional media information (e.g., images) to assist subjects. |
| **Data Collection Elements** | |
| *4. Question Types* | |
| 4.1. DropDown | Only one item may be selected. |
| 4.2. Single Choice | Only one item may be selected. |
| 4.3. Yes No Switch | Only one item may be selected. |
| 4.4. Range | Multiple items may be selected. |
| 4.5. Multiple Choice | Multiple items may be selected. |
| 4.6. Ranking | Items may be ordered according to own preferences. |
| 4.7. Distribution | Points may be spent among available items. |
| 4.8. Slider | One value from a scale may be selected. |
| 4.9. Freetext | Answer using regular text input (text, number, date). |

the respective elements may be edited. Note that the configurator allows handling multiple languages as well as keeps track of different element revisions. Finally, created elements may be combined to pages using drag and drop operations. It also provide an interactive live preview of the modeled element (or page) in order to offer an immediate feedback for healthcare experts. Note that the preview takes the configured languages as well as different smart mobile devices (e.g., smartphones or tablets) into account.

**Modeling Area View** (see Fig. 4, right). Healthcare experts may combine the previously created pages in order to model the data collection instrument by applying simple drag & drop operations. Furthermore, they are able to model sophisticated navigation operations to provide guidance during the data collection process. The graphical editor, in turn, strictly follows a correctness-by-construction approach; i.e., it is ensured that created models are executable by the lightweight process engine that runs on heterogeneous smart mobile devices. When deploying the model to respective smart mobile devices, it is automatically mapped to an executable process model according to the previously introduced approach.

The configurator allows creating all elements needed to design a data collection instrument (e.g., texts or questions). The latter may be provided in different languages to enable multilingualism. Through its model-driven development approach healthcare experts are enabled to easily define the logic and structure of the data collection instrument themselves. Advanced wizards guide healthcare experts through the process of defining navigation paths for instruments. Finally, sensors for collecting data during run time (e.g., vital parameters of patients) are modeled on an abstract level.

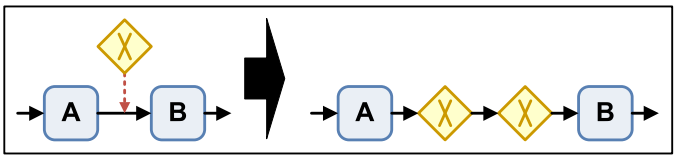
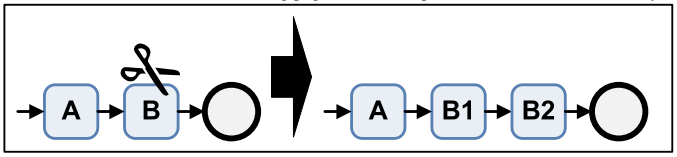
**TABLE 3.** Identified change patterns.

| | Tinnitus Research | Risk Factors during Pregnancy | Adverse Childhood Experiences | PTSD in War Regions | Learning Deficits among Students | Supporting Children after Accidents |
|---|---|---|---|---|---|---|
| S1 Insert Page | >5 | >25 | >80 | >150 | >10 | >5 |
| S2 Insert Block | 0 | >15 | >80 | >100 | >15 | >5 |
| S3 Insert Null Path | 0 | >15 | >90 | >250 | >20 | >5 |
| S4 Embed Instrument | 0 | 1 | 1 | >20 | 0 | 0 |
| C1 Insert Element | >60 | >120 | >200 | >1500 | >100 | >40 |
| C2 Move Element | >10 | >40 | >50 | >500 | >20 | >10 |
| R1 Cut Page | 0 | >5 | >10 | >30 | 0 | 0 |
| R2 Merge Page | 0 | >5 | >10 | >15 | 0 | 0 |
| R3 Move Page | 0 | >10 | >15 | >60 | 0 | 0 |
| R4 Update Decision | 0 | >5 | >30 | >80 | >15 | 0 |

### 1) DATA COLLECTION INSTRUMENT REFINEMENT PATTERNS

When developing various mobile data collection applications based on traditional paper-based instruments, recurring operations could be identified. These so-called *change patterns* allow healthcare experts to easily create and adapt existing mobile data collection instruments to new requirements. Note that the patterns were derived by evaluating more than 40 instruments from different healthcare fields (see Table 3; estimated values, as the projects are still ongoing). The patterns particularly facilitate the model-driven development technique of constantly refining models until proper code fragments (i.e., a proper executable process model)

**TABLE 4.** Selected refinement patterns.

| | |
|---|---|
| **Name** | Insert Block. Add a new block to an existing instrument (between `before` and `after`). Available `types` are `IF`, `ALL`, and `REPEAT`. |
| **Signature** | `insertBlock(type, before, after)` |
| **Example** | Depending on the `type` of the block, various scenarios are possible:<br>• `IF` blocks solely select *one path* based on already given answers during run time.<br>• `ALL` blocks select *all paths* to be executed, however, the person interacting with the smart mobile device may choose its order of execution.<br>• `REPEAT` blocks allow for repeating the content of the block multiple times. The amount of repetitions may be determined at run time (e.g., based on given answers) or are pre-defined by the domain expert (e.g., $n$ times).<br>  |
| **Pre-Condition** | The position to insert the `block` must be exactly specified; i.e., `after` must directly follow `before`. |
| **Post-Condition** | An empty block comprising a `split` and `join` gateway that are directly connected is inserted; For `IF` and `REPEAT` blocks, data elements for evaluating the conditions need to be connected using `READ` data edges. |
| **Name** | Cut Page. Cut a `page` after the given `position` in order to split the latter in two pages. |
| **Signature** | `cutPage(page, position)` |
| **Example** | A domain expert wants to split a `page` in two in order to enhance readability of the instrument or utilize the device screen more properly. Furthermore, the order of respective elements within the two resulting pages must be kept in order to ensure the validity of the data collection instrument.<br>  |
| **Pre-Condition** | The `page` to be cut in half must not be an `instrument`. The `position` to cut the page must be valid (i.e., $0 <$ `position` $<$ `size(elementsInPage)`) |
| **Post-Condition** | The initial `page` is removed, both resulting `pages` must not be empty (i.e., they contain elements). Furthermore, all `data elements` are correctly assigned to either one of the two `pages`. |

for a specific scenario can be derived. These patterns are assigned to different levels, reflecting specific aspects of the data collection instrument. Patterns of the first level solely correspond to the *structure* (e.g., the flow) of the instrument, whereas patterns of the second level refer to the *content* of pages. Finally, *refactoring* patterns enable healthcare experts to change aspects of the instruments, while still maintaining respective validity.

**Structural Change Patterns (S)** provide features to create and maintain the logic of a data collection instrument. These patterns include, for example, adding *pages* or *blocks* in order to provide sophisticated navigation operations.

**Content Change Patterns (C)** enable the management of *elements* (e.g., headlines or questions; see Table 2) within a specific page. Data elements for capturing answers, as well as their corresponding data edges, are automatically created when using these patterns.

**Refactoring Change Patterns (R)** allow modelers to adapt an instrument to new requirements without violating validity constraints. For example, a page containing demographic questions may be moved, while other pages referring to these questions are updated accordingly.

The combined use of the identified change patterns allows healthcare experts to create and adapt mobile data collection instruments in a more flexible manner. In addition, the change patterns reflect the technique of constantly refining a model as it is a basic pillar of any model-driven development. By providing refactoring patterns, they additionally foster the continuous development of instruments. Fig. 5 illustrates

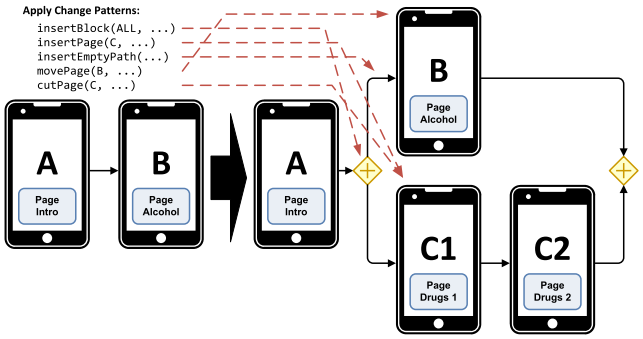



**FIGURE 5.** Applying change patterns to a data collection instrument.

how a set of change patterns may adapt an existing mobile data collection application to new requirements. Note that the presented configurator component provides high-level access for the introduced patterns. Finally, all identified change patterns are described using a graphical representation and an example from considered real-world scenarios. Moreover, pre- and post-conditions for applying the patterns are shown (see Table 4).

### B. DEPLOYING A PLATFORM INDEPENDENT MODEL

In order to collect data using the modeled data collection instrument, the latter is deployed on a centralized server component. This web service, in turn, offers RESTful routes to (1) upload the instruments and (2) download the process model as well as (3) to provide required configuration data

and meta information on clients (e.g., smart mobile devices). Note that the deployment process can be triggered automatically from the configurator component. After manually modeling the instrument using the process-aware configurator, the healthcare expert is able to upload it on the intermediary service. However, before uploading the latter, it is transformed to an executable process model (as described in Section V) and the *platform independent model* is generated. Smart mobile devices are now able to install (i.e., download) and import the instrument into their locally running process engine in order to create new instances of the instrument. The process engine as well as the installed *executable components*, in turn, act as *platform specific model* and *code model* known from the *model-driven development idea* (see Section IV).

### C. EXECUTING A PLATFORM SPECIFIC CODE MODEL

In order to collect data, the instrument is instantiated and enacted using a lightweight process engine that runs directly on smart mobile devices. Furthermore, this engine ensures the robust execution based on the created model as well as the specified behavior of its control flow elements (e.g., `XORsplit`, or `LOOP` blocks).

Reference [9] describes in detail how a smart mobile application integrates the developed lightweight mobile process engine in order to collect data using a well-defined model. For example, a new instance of an instrument is created by invoking the `Execution Manager` of the mobile process engine. The `Instance Manager` then validates the current status (e.g., all data elements are correctly set) and activates the next node (i.e., page of an instrument). Such a node, thereby, contains both, platform independent information for displaying itself as well as platform specific information how to properly process the latter. This information is handed over to the `Runtime Manager`, which passes *input variables* to so-called *executable components* (`EC`). These components serve as basis for dynamically extending the provided functionality of the developed mobile data collection application. An `EC` can be seen as *Micro Service* [59] that provides complex logic how to render a user interface or calculate the further progress of the instrument. Finally, the rendered user interface is returned and integrated in the overall user interface of the mobile data collection application. Note that `ECs` are not part of the data collection application or the lightweight process engine, but are rather installed as specific applications on the smart mobile device. This allows for updating a specific `EC` independently, in order to change its functionality or user-interface, or install novel `ECs` to add features to the data collection application on the fly. The mobile process engine and its executable components, in turn, act as the *platform specific model* for the enactment. During the enactment of an instrument (i.e., the process model), it gets transformed to the *code model* that is executed by an user interacting with the smart mobile device.

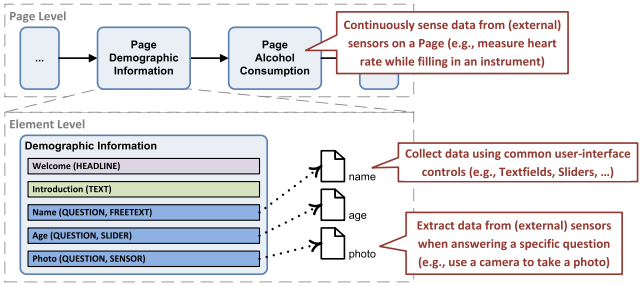



**FIGURE 6.** Sensing data on different levels.

### D. SENSING DATA ON DIFFERENT LEVELS

Two approaches to sense data have to be distinguished in QuestionSys. First, sensing data of an instrument is solely accomplished by entering data using the different questionnaire elements. Note that instruments that are represented by questionnaires inherently provide sensing capabilities (e.g., by changing the order of questions also the sensing is adjusted). Second, internal and external sensors of a smart mobile device can be added to the data collection instrument. Based on these two sensing approaches and the model-driven development techniques, the QuestionSys framework provides a sophisticated sensing of data on different levels (see Fig. 6).

First of all, the *Element Level* uses common user-interface controls for entering data, like textfields, sliders or checkboxes (see Table 2). Furthermore, it is possible to configure (external) *sensors* that are connected to the smart mobile device in order collect additional information. For example, the `Photo` question (within the `Demographic Information` page) allows for accessing the device's camera in order to take a photo of the participant. Note that these types of sensor values are specifically bound to a given question and represent a snapshot of the current situation. In addition, it is possible to combine different sensors within one page. In specific application scenarios, in turn, it may be required to collect data on a continuous basis. Therefore, the QuestionSys framework allows defining sensors on the *Page Level* as well. However, this requires sensors to support continuous data collection (e.g., measure the current heart rate per second). The described configurator component enables healthcare experts to add sensors to elements or pages. Thereby, a sensor is a specific type of question with an additional configuration that contains all required information to connect the sensor to the device (e.g., the protocol they are using (Bluetooth)), or the frequency to collect data (e.g., measure the heart rate every second). Thereby, implementing the code to actually retrieve data from a sensor and processing its data has to be implemented by application developers. However, through the usage of the executable components, the process model, and the model-driven development, sensors can be easily used. Altogether, QuestionSys provides features to integrate internal and external sensors of smart mobile devices to mobile data collection applications. Moreover, the way instruments

are generally supported paves the way to beneficial sensing opportunities.

## VII. STUDY RESULTS

In order to validate the perceived complexity of the proposed framework, a controlled study, involving more than 40 participants, was conducted [11]. In particular, the study tackled the question whether experts understand the modeling concept with respect to the complexity of the configurator component (see Section VI-A). At the beginning of the study, each participant had to fill in a demographic questionnaire, which mainly gathered information on personal data, literacy, and prior process modeling experience. Based on the latter information, the participants were divided into two groups, namely *experts* and *novices*. During the modeling of instruments, the configurator logged data regarding the usage of the application, the *time needed* to complete a task, or the *errors made* during modeling. Furthermore, questions regarding their mental effort when using the configuration had to be answered. When analyzing the results, some kind of *learning effect* could be observed between the different tasks when specifically looking at the novices – the errors made between Task 1 and 2 dropped dramatically (i.e., Task 1: 4 errors; Task 2: 1 error), while experts tend to be the same (i.e., Task 1: 1 error; Task 2: 1 error). This shows that prior process modeling experience has minimal effect on the overall understanding of the configurator component. Note that none of the participants had worked with the application before and all gained respective knowledge to use the configurator component properly in an adequate time (approx. 1h duration of the study). Based on this pilot study, we conducted a larger study with 80 participants not involved in the first study before, with an even more sophisticated study design that comprises two testing sessions (second session 7 days after the first one). All materials and methods were approved by the Ethics Committee of Ulm University and were carried out in accordance with the approved guidelines (cf. [12]). All participants gave their informed consent. Within each session, participants had to model 5 data collection instruments (10 in total). As described before, the configurator logged basic performance measures. Again, participants were divided into two groups (i.e., *novices* and *experts* respectively) based upon their prior knowledge in process modeling. Note that this is only one (simplified) possibility to classify participants into those groups. Another possibility would be more in-depth questioning of participants (e.g., asking about familiarity with notations such as BPMN, asking for examples of process models they have created, and asking specific questions about particular items in process modeling notations). This would lead to a spectrum of rated expertise, rather than the simplified binary approach used in this study. Moreover, the following baseline differences emerged [12]. The novices' sample contained more female participants, whereas the experts' sample contained male participants ($p < .05$). In general, the experts' sample had more participants with Bachelor as highest education level than the novices' sample. The novices, however, had a larger amount with High School graduates ($p < .05$). Finally, these samples also differ in their field of study; the vast majority of novices studied Psychology, whereas the vast majority of experts studied Economics or Computer Science ($p < .05$). We performed 2 established tests measuring the participants' processing speed in order to proof similar cognitive abilities in both groups. That means, for example, although experts have a higher education and were older than novices, their cognitive abilities did not differ. This study also evaluated the participants' perceived complexity of the modeling task compared to three assessed performance measures (operations, time, and errors). After modeling each instrument, participants had to subjectively evaluate if they were able to solve the respective task properly (higher values indicate more complexity). Thereby, the following research questions (RQ) were addressed by inferential tests performed two-tailed with a significance value of $p < .05$:

### A. RQ1: HOW ARE PERFORMANCE MEASURES OF NOVICES AND EXPERTS COMPARED TO THE PERCEIVED COMPLEXITY OF EACH DATA COLLECTION INSTRUMENT?

Pearson correlation coefficients were computed to investigate associations between performances measures and subjective complexity for the novices' as well as for the experts' sample. In the novices' sample, perceived complexity correlated positively and significantly with the performance measures 14 times (in 30 comparisons), whereas the perceived complexity correlated significantly with the performance measures 12 times (in 30 comparisons again) in the experts' samples (see Table 5). Therefore, more operations, more time, and more errors were associated with more subjective complexity only in less than 50% of all comparisons.

### B. RQ2: HOW ARE PERFORMANCE MEASURES OF NOVICES COMPARED TO THE PERCEIVED COMPLEXITY ACROSS ALL DATA COLLECTION INSTRUMENTS?

Linear multilevel models with two levels (data collection instruments nested within participants) were performed to evaluate associations between the performance measures and subjective complexity across all data collection instruments for novices. In the following results, the intercept stands for the estimated value of the performance measure when statistically controlling for subjective complexity.

*Operations:* Intercept was 9.92 ($SE = .61$) and reached statistical significance ($T(445) = 16.14$; $p < .001$). Increases on the perceived complexity scale were associated with an increase in operations: 1.21 ($SE = .22$) per perceived complexity point and this was significant ($T(445) = 5.44$; $p < .001$).

*Time:* Intercept was 121824.93 (measured in msec; $SE = 15995.44$) and reached statistical significance ($T(445) = 7.62$; $p < .001$); increases on the perceived complexity scale were associated with an increase in time: 33794.88 (measured in msec; $SE = 5766.46$) per perceived complexity point and this was significant ($T(445) = 5.86$; $p < .001$).

**TABLE 5.** Correlations between perceived complexity and performance measures for novices and experts.

| Tasks | Session | Novices | | | Experts | | |
|---|---|---|---|---|---|---|---|
| | | Operations | Time | Errors | Operations | Time | Errors |
| 1 | 1 | .344* | .475** | .486** | .435** | .410* | .621** |
| 2 | 1 | .350* | .532** | .613** | .448** | .550** | .391* |
| 3 | 1 | .035 | .093 | .233 | .534** | .457** | .362* |
| 4 | 1 | .042 | .059 | .524** | .299 | .357* | .058 |
| 5 | 1 | .187 | .028 | .495** | .299 | .206 | .445** |
| 6 | 2 | .289 | .307* | .257 | .329 | .380* | -.073 |
| 7 | 2 | .174 | .269 | .362* | .162 | .198 | .113 |
| 8 | 2 | .323* | .054 | .467** | .199 | .258 | .258 |
| 9 | 2 | .252 | .207 | .372* | .306 | .201 | .226 |
| 10 | 2 | .119 | .100 | .506** | -.124 | .011 | .039 |

with $* = p < .05$ and $** = p < .01$.

*Errors:* Intercept was $-.50$ ($SE = .16$) and reached statistical significance ($T(445) - 3.14$; $p = .002$); increases on the perceived complexity scale were associated with an increase in errors: .60 ($SE = .06$) per perceived complexity point and this was significant ($T(445) = 10.49$; $p < .001$).

In summary, increases on the subjective complexity scale were associated with worse performance measures in the novices' samples across all data collection instruments.

### C. RQ3: HOW ARE PERFORMANCE MEASURES OF EXPERTS COMPARED TO THE PERCEIVED COMPLEXITY ACROSS ALL DATA COLLECTION INSTRUMENTS?

Again, linear multilevel models with two levels (data collection instruments nested within participants) were performed to evaluate associations between the performance measures and subjective complexity across all data collection instruments for experts. The intercept indicates again the estimated value of the performance measure when statistically controlling for subjective complexity.

*Operations:* Intercept was 7.04 ($SE = .69$) and reached statistical significance ($T(340) = 10.19$; $p < .001$); increases on the perceived complexity scale were associated with an increase in operations: 2.52 ($SE = .31$) per perceived complexity point and this was significant ($T(340) = 8.20$; $p < .001$).

*Time:* Intercept was 59191.80 (measured in msec; $SE = 17249.28$) and reached statistical significance ($T(340) = 3.43$; $p = .001$); increases on the perceived complexity scale were associated with an increased time: 63038.41 (measured in msec; $SE = 7664.68$) per perceived complexity point and this was significant ($T(340) = 8.23$; $p < .001$).

*Errors:* Intercept was $-.17$ ($SE = .09$) and did not reach statistical significance ($T(335) = -1.75$; $p = .081$); increases on the perceived complexity scale were associated with an increase in errors: .25 ($SE = .04$) per perceived complexity point and this was significant ($T(335) = 5.61$; $p = .001$).

As in the novices' sample, increases on the subjective complexity scale were also correlated with worse performance measures in the experts' sample across all data collection instruments.

Altogether, both studies indicate a promising perspective of the developed framework. Furthermore, the latter constitutes a suitable approach for healthcare experts with no programming knowledge to develop sophisticated mobile applications for collecting data in various scenarios. Finally, the QuestionSys approach may act as benchmark for collecting data using smart mobile devices in general.

To conclude to contribution C2 (see Section I), the study results (i.e., based on the perceived complexity and mental effort) indicate that healthcare experts are able to properly create mobile data collection applications on their own without the involvement of IT experts.

## VIII. SUMMARY AND OUTLOOK

Based on the realized projects, the important aspects required for a generic solution that is able to provide mobile data collection applications for healthcare settings have been presented in this paper. While some of them are related to the actual development and realization of mobile applications, most of them, however, arise from communication problems between healthcare experts and application developers. In order to deal with these issues, an approach was presented that combines process management technology with a model-driven development in a much broader scope. The combined use of these technologies enable healthcare experts to create mobile data collection applications themselves, reducing costs and time for its implementation. Hence, the exploitation of the increased capabilities of smart mobile devices becomes possible. In this context, the presented change patterns and the different levels of sensing allow for a rapid creation and adaptation of mobile data collection applications to meet field-specific requirements from healthcare appropriately. Results of conducted studies were presented that show that QuestionSys goes beyond a technical prototype. In order to strengthen the results of the studies, as well as to remove possible limitations, further healthcare studies are currently conducted. They specifically focus on the observed learning effect and measure mental efforts over a longer period of time and with a higher precision. Furthermore, we are working on *quality metrics* that allow for mutually comparing modeled instruments. Thereby, model-based metrics as well as common software code metrics are applied in order to estimate the mental effort or time needed to properly process this instrument. These metrics finally aim at supporting healthcare experts in the decision-making progress whether or not

an instrument fits to a specific scenario. Despite the powerful features of QuestionSys, healthcare experts often requested a feature that guides them with recommendation criteria when creating completely new instruments. This issue will be tackled in future work. Altogether, the proposed approach may significantly change the way mobile data collection applications are created. This has been shown by providing details on the contributions C1 & C2 (see Section I). We believe that especially the healthcare domain as well as the life-sciences in general will benefit from our approach. However, other domains more and more crave for using smart mobile devices in everyday and business life to collect data in a new way.

## REFERENCES

[1] R. Fernandez-Ballesteros, "Self-report questionnaires," in *Comprehensive Handbook of Psychological Assessment*, vol. 3, 2004, pp. 194–221.
[2] I. Pavlovič, T. Kern, D. Miklavčič, "Comparison of paper-based and electronic data collection process in clinical trials: Costs simulation study," *Contemp. Clin. Trials*, vol. 30, no. 4, pp. 300–316, 2009.
[3] P. Carlbring *et al.*, "Internet vs. paper and pencil administration of questionnaires commonly used in panic/agoraphobia research," *Comput. Hum. Behav.*, vol. 23, no. 3, pp. 1421–1434, 2007.
[4] T. M. Palermo, D. Valenzuela, and P. P. Stork, "A randomized trial of electronic versus paper pain diaries in children: Impact on compliance, accuracy, and acceptability," *Pain*, vol. 107, no. 3, pp. 213–219, 2004.
[5] S. J. Lane, N. M. Heddle, E. Arnold, and I. Walker, "A review of randomized controlled trials comparing the effectiveness of hand held computers with paper methods for data collection," *BMC Med. Inform. Decis. Making*, vol. 6, no. 1, p. 23, 2006.
[6] J. A. Ellis, "Leveraging mobile phones for monitoring risks for noncommunicable diseases in the future," *J. Med. Internet Res.*, vol. 19, no. 5, p. e137, 2017.
[7] J. Schobel, R. Pryss, M. Schickler, M. Ruf-Leuschner, T. Elbert, and M. Reichert, "End-user programming of mobile services: Empowering domain experts to implement mobile data collection applications," in *Proc. IEEE 5th Int. Conf. Mobile Services*, Jun./Jul. 2016, pp. 1–8.
[8] J. Schobel, M. Schickler, R. Pryss, H. Nienhaus, and M. Reichert, "Using vital sensors in mobile healthcare business applications: Challenges, examples, lessons learned," in *Proc. 9th Int. Conf. Web Inf. Syst. Technol.*, May 2013, pp. 1–10.
[9] J. Schobel, R. Pryss, M. Schickler, and M. Reichert, "A lightweight process engine for enabling advanced mobile applications," in *Proc. 24th Int. Conf. Cooperat. Inf. Syst.* Lecture Notes in Computer Science, vol. 10033. Springer, Oct. 2016, pp. 552–569.
[10] J. Schobel, R. Pryss, M. Schickler, and M. Reichert, "A configurator component for end-user defined mobile data collection processes," in *Proc. 14th Int. Conf. Service-Oriented Comput.*, Oct. 2016, pp. 216–219.
[11] J. Schobel *et al.*, "Development of mobile data collection applications by domain experts: Experimental results from a usability study," in *Proc. 29th Int. Conf. Adv. Inf. Syst. Eng. (CAiSE)* Lecture Notes in Computer Science, vol. 10253. Springer, Jun. 2017, pp. 60–75.
[12] J. Schobel, R. Pryss, T. Probst, W. Schlee, M. Schickler, and M. Reichert, "Learnability of a configurator empowering end users to create mobile data collection instruments: Usability study," *JMIR mHealth uHealth*, vol. 6, no. 6, p. e148, 2018.
[13] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Reading, MA, USA: Addison-Wesley, 2003.
[14] R. Soley *et al.*, "Model driven architecture," OMG White Paper 308, 308, 5, 2000.
[15] A. W. Brown, "Model driven architecture: Principles and practice," *Softw. Syst. Model.*, vol. 3, no. 4, pp. 314–327, 2004.
[16] D. Wolber, "App inventor and real-world motivation," in *Proc. 42nd ACM Tech. Symp. Comput. Sci. Edu.* New York, NY, USA: ACM, 2011, pp. 601–606.
[17] A. Ribeiro and A. R. da Silva, "Evaluation of xis-mobile, a domain specific language for mobile application development," *J. Softw. Eng. Appl.*, vol. 7, no. 11, pp. 906–919, 2014.
[18] F. Balagtas-Fernandez, M. Tafelmayer, and H. Hussmann, "Mobia Modeler: Easing the creation process of mobile applications for non-technical users," in *Proc. 15th Int. Conf. Intell. User Interfaces*. New York, NY, USA: ACM, 2010, pp. 269–272.
[19] H. Heitkötter, T. A. Majchrzak, and H. Kuchen, "Cross-platform model-driven development of mobile applications with $md^2$," in *Proc. 28th Annu. ACM Symp. Appl. Comput.* New York, NY, USA: ACM, 2013, pp. 526–533.
[20] A. H. Ranabahu, E. M. Maximilien, A. P. Sheth, and K. Thirunarayan, "A domain specific language for enterprise grade cloud-mobile hybrid applications," in *Proc. Compilation Co-Located Workshops DSM, TMC, AGERE! AOOPES, NEAT, VMIL*. New York, NY, USA: ACM, 2011, pp. 77–84.
[21] O. Le Goaer and S. Waltham, "Yet another dsl for cross-platforms mobile development," in *Proc. 1st Workshop Globalization Domain Specific Lang.* New York, NY, USA: ACM, 2013, pp. 28–33.
[22] A. Khambati, J. Grundy, J. Warren, and J. Hosking, "Model-driven development of mobile personal health care applications," in *Proc. 23rd IEEE/ACM Int. Conf. Automated Softw. Eng.*, Sep. 2008, pp. 467–470.
[23] J. Dunkel and R. Bruns, "Model-driven architecture for mobile applications," in *Proc. Int. Conf. Bus. Inf. Syst.* Springer, 2007, pp. 464–477.
[24] M. Brambilla and P. Fraternali, "Large-scale model-driven engineering of web user interaction: The webml and webratio experience," *Sci. Comput. Program.*, vol. 89, pp. 71–87, Sep. 2014.
[25] M. Henkel and J. Stirna, "Pondering on the key functionality of model driven development tools: The case of mendix," in *Perspectives in Business Informatics Research*, 2010, pp. 146–160.
[26] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 402–427, 1st Quart., 2013.
[27] J. Liu, H. Shen, H. S. Narman, W. Chung, and Z. Lin, "A survey of mobile crowdsensing techniques: A critical component for the Internet of Things," *ACM Trans. Cyber-Phys. Syst.*, vol. 2, no. 3, p. 18, 2018.
[28] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, p. 7, 2015.
[29] A. Seifert, M. Hofer, and M. Allemand, "Mobile data collection: Smart, but not (yet) smart enough," *Frontiers Neurosci.*, vol. 12, p. 971, Dec. 2018.
[30] N. Bashi, F. Fatehi, M. Fallah, D. Walters, and M. Karunanithi, "Self-management education through mhealth: Review of strategies and structures," *JMIR mHealth uHealth*, vol. 6, no. 10, 2018, Art. no. e10771.
[31] P.-Y. Hsueh, P. Melville, and V. Sindhwani, "Data quality from crowdsourcing: A study of annotation selection criteria," in *Proc. NAACL HLT Workshop Act. Learn. Natural Lang. Process.* Assoc. Comput. Linguistics, 2009, pp. 27–35.
[32] F. Restuccia, N. Ghosh, S. Bhattacharjee, S. K. Das, and T. Melodia, "Quality of information in mobile crowdsensing: Survey and research challenges," *ACM Trans. Sensor Netw.*, vol. 13, no. 4, p. 34, 2017.
[33] D. D. Luxton, R. A. McCann, N. E. Bush, M. C. Mishkind, and G. M. Reger, "mhealth for mental health: Integrating smartphone technology in behavioral healthcare," *Prof. Psychol., Res. Pract.*, vol. 42, no. 6, p. 505, 2011.
[34] A. Gaggioli *et al.*, "A mobile data collection platform for mental health research," *Pers. Ubiquitous Comput.*, vol. 17, no. 2, pp. 241–251, 2013.
[35] R. Pryss, N. Mundbrod, D. Langer, and M. Reichert, "Supporting medical ward rounds through mobile task and process management," *Inf. Syst. e-Bus. Manage.*, vol. 13, no. 1, pp. 107–146, Feb. 2015.
[36] A. Vankipuram, M. Vankipuram, V. Ghaemmaghami, and V. L. Patel, "A mobile application to support collection and analytics of real-time critical care data," *Comput. Methods Programs Biomed.*, vol. 151, pp. 45–55, Nov. 2017.
[37] M. Zhang, E. Cheow, C. S. Ho, B. Y. Ng, R. Ho, and C. C. S. Cheok, "Application of low-cost methodologies for mobile phone app development," *JMIR mHealth uHealth*, vol. 2, no. 4, p. e55, 2014.
[38] M. W. Zhang, T. Tsang, E. Cheow, C. S. Ho, N. B. Yeong, and R. C. Ho, "Enabling psychiatrists to be mobile phone app developers: Insights into app development methodologies," *JMIR mHealth uHealth*, vol. 2, no. 4, p. e53, 2014.
[39] S. Kim, J. Mankoff, and E. Paulos, "Sensr: Evaluating a flexible framework for authoring mobile data-collection tools for citizen science," in *Proc. Conf. Comput. Supported Cooperat. Work*. New York, NY, USA: ACM, 2013, pp. 1453–1462.
[40] M. Cavelaars *et al.*, "OpenClinica," in *Journal of Clinical Bioinformatics*, vol. 5. Springer, 2015, p. S2.

[41] C. Gao, F. Kong, and J. Tan, "Healthaware: Tackling obesity with health aware smart phone systems," in *Proc. IEEE Int. Conf. Robot. Biomimetics. (ROBIO)*, Dec. 2009, pp. 1549–1554.
[42] W.-J. Yi, W. Jia, and J. Saniie, "Mobile sensor data collector using android smartphone," in *Proc. IEEE 55th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2012, pp. 956–959.
[43] K. Sha, G. Zhan, W. Shi, M. Lumley, C. Wiholm, and B. Arnetz, "SPA: A smart phone assisted chronic illness self-management system with participatory sensing," in *Proc. 2nd Int. Workshop Syst. Netw. Support Health Care Assist. Living Environ.* New York, NY, USA: ACM, 2008, p. 5.
[44] M. Schickler, M. Reichert, R. Pryss, J. Schobel, W. Schlee, and B. Langguth, *Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health* (eXamen.press). Springer Vieweg, 2015.
[45] P. Adamczyk, P. H. Smith, R. E. Johnson, and M. Hafiz, "REST and Web services: In theory and in practice," in *REST: From Research to Practice*. Springer, 2011, pp. 35–57.
[46] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. big'web services: Making the right architectural decision," in *Proc. 17th Int. Conf. World Wide Web*. New York, NY, USA: ACM, 2008, pp. 805–814.
[47] R. Pryss, J. Schobel, and M. Reichert, "Requirements for a flexible and generic API enabling mobile crowdsensing mhealth applications," in *Proc. 4th Int. Workshop Requirements Eng. Self-Adapt., Collaborative, Cyber Phys. Syst. (RESACS)*, Aug. 2018, pp. 24–31.
[48] R. Mulero *et al.*, "An AAL system based on IoT technologies and linked open data for elderly monitoring in smart cities," in *Proc. 2nd Int. Multidisciplinary Conf. Comput. Energy Sci. (SpliTech)*, Jul. 2017, pp. 1–6.
[49] A. W. Brown, J. Conallen, and D. Tropeano, "Introduction: Models, modeling, and model-driven architecture (MDA)," in *Model-Driven Software Development*. Springer, 2005, pp. 1–16.
[50] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. London, U.K.: Pearson Higher Education, 2004.
[51] M. Reichert and P. Dadam, "ADEPT$_{flex}$–Supporting dynamic changes of workflows without losing control," *J. Intell. Inf. Syst.*, vol. 10, no. 2, pp. 93–129, 1998.
[52] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2010.
[53] R. France and B. Rumpe, "Model-driven development of complex software: A research roadmap," in *Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 37–54.
[54] J. Schobel, R. Pryss, W. Wipp, M. Schickler, and M. Reichert, "A mobile service engine enabling complex data collection applications," in *Proc. 14th Int. Conf. Service Oriented Comput.* Lecture Notes in Computer Science, vol. 9936, Oct. 2016, pp. 626–633.
[55] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Berlin, Germany: Springer, 2012.
[56] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Upper Saddle River, NJ, USA: Prentice-Hall, 2005.
[57] D. Guinard, I. Ion, and S. Mayer, "In search of an Internet of Things service architecture: REST or WS-*? A developers' perspective," in *Proc. Int. Conf. Mobile Ubiquitous Syst., Comput., Netw., Services*. Springer, 2011, pp. 326–337.
[58] D. Ruiz-Fernández, D. Marcos-Jorquera, V. Gilart-Iglesias, V. Vives-Boix, and J. Ramírez-Navarro, "Empowerment of patients with hypertension through BPM, iot and remote sensing," *Sensors*, vol. 17, no. 10, p. 2273, 2017.
[59] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. Newton, MA, USA: O'Reilly Media, 2015.

**JOHANNES SCHOBEL** studied computer science at Ulm University. He completed the Ph.D. thesis, in 2018. He has been with the Institute of Databases and Information Systems as a Research Associate, since 2012. His main research focus is on mobile data collection. In particular, he focuses on end-user programming approaches to empower domain experts to create their own mobile data collection applications. In this context, he applies business process management techniques and end-user programming approaches to unravel new insights.

**THOMAS PROBST** studied Psychology at Regensburg University. He holds a Diploma in psychology. After graduating, he started his psychotherapy training and received his certification as cognitive-behavior therapist in 2013. Between 2013 and 2015, he worked at Regensburg University (as a research assistant and deputy head of the psychotherapy outpatient center). In 2015, he received the Ph.D. degree in psychology from the Humboldt-University of Berlin. In his doctoral thesis, Thomas focused on psychotherapy monitoring, patient–therapist feedback, and decision support tools. From 2015 to 2016, he was Interim Professor for Clinical Psychology and Psychotherapy, as well as for Clinical Psychodiagnostics at the University Witten/Herdecke. In 2017, he was Interim Professor at the Georg-August-University Göttingen and research associate at Ulm University. At 2017, he was appointed as Professor for Psychotherapy Sciences at the Danube University Krems, Austria. Moreover, he is experienced with teaching courses on psychotherapy and psychodiagnostics, psychosomatics, digital health, quantitative research designs.

**MANFRED REICHERT** received the Ph.D. degree in computer science and the Diploma degree in mathematics. He was an Associate Professor with the University of Twente, The Netherlands, where, he was also a member of the Management Board of the Centre for Telematics and Information Technology, which is one of the largest academic ICT research institutes in Europe. Since 2008, he has been a Full Professor with the University of Ulm, where he is the Director of the Institute of Databases and Information Systems. His research interests include business process management (e.g., adaptive and flexible processes, process lifecycle management, and data-driven and object-centric processes) and service-oriented computing (e.g., service interoperability, mobile services, and service evolution). He has been the PC Co-chair of the BPM'08, CoopIS'11, EMISA'13, and EDOC'13 conferences, and the General Chair of the BPM'09 and EDOC'14 conferences.

**MARC SCHICKLER** studied computer science at Ulm University. He completed the Ph.D. thesis, in 2018. He has been with the Institute of Databases and Information Systems as a Research Associate, since 2012. His main research focus is on mobile therapeutic interventions. Particularly, he focuses on the seamless integration of modern smart mobile devices (e.g., smartphones and wearables) into different therapy scenarios. Thereby, he exploits business process management technology to configure therapeutic homework, on the one hand, and execute them on the patient's smart mobile device on the other.

**RÜDIGER PRYSS** studied at the Universities of Passau, Karlsruhe and Ulm. He holds a Diploma in Computer Science. After graduating, he worked as a consultant and developer in a software company. Since 2008, he has been a research associate at Ulm University. In 2015, he received the Ph.D. degree in Computer Science. In his doctoral thesis, he focused on fundamental issues related to mobile process and task support. He was local organization chair of the BPM'09 and EDOC'14 conferences. Moreover, he is experienced with teaching courses on database management, programming, service-oriented computing, business process management, document management, and mobile application engineering. Currently, Rüdiger Pryss finishes his habilitation at Ulm University.