A Mobile Service Engine Enabling Complex Data Collection Applications

Johannes Schobel(✉), Rüdiger Pryss, Wolfgang Wipp, Marc Schickler, and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Ulm, Germany
{johannes.schobel,ruediger.pryss,wolfgang.wipp,marc.schickler, manfred.reichert}@uni-ulm.de

Abstract. The widespread distribution of smart mobile devices offers promising perspectives for the timely collection of huge amounts of data. When realizing sophisticated mobile data collection applications, numerous technical issues arise. For example, as many real-world projects require the support of different mobile operating systems, platform-specific peculiarities must be properly handled. Existing approaches often rely on specifically tailored mobile applications. As a drawback, changes to the data collection procedure result in costly code adaptations. To remedy this drawback, a model-driven approach is proposed, enabling end-users (i.e., domain experts) to create mobile data collection applications themselves. This model relies on complex questionnaires called instruments. An instrument not only contains all information about the data to be collected, but additionally comprises information on how it shall be processed on different mobile operating systems. For this purpose, we developed an advanced mobile (kernel) service being capable of processing sophisticated instruments on various platforms. This paper discusses fundamental kernel requirements and introduces the developed architecture. Altogether, the mobile service allows for the effective use of smart mobile devices in data collection application scenarios (e.g., clinical trials).

Keywords: Mobile process engine · Mobile data collection · Smart mobile device · Mobile process · Mobile healthcare

1 Introduction

Smart mobile devices are increasingly used in everyday life. In line with this trend, application domains for which huge amounts of data must be collected (e.g., clinical trials) significantly benefit from using mobile applications. These scenarios range from fitness trackers to applications monitoring vital parameters. However, when realizing mobile data collection applications, profound knowledge from real-world scenarios is essential.

In various large-scale mobile data collection applications we realized (cf. Table 1), domain experts (e.g., medical doctors) were provided with specifically

Q.Z. Sheng et al. (Eds.): ICSOC 2016, LNCS 9936, pp. 626–633, 2016.
DOI: 10.1007/978-3-319-46295-0_42

Table 1. Realized mobile data collection applications

#	Data collection applications	Country	CN	Releases	Instances
1	Tinnitus research	World-Wide	○	3	≥20,000
2	Risk factors during pregnancy	Germany	○	5	≥1,000
3	Risk factors after pregnancy	Germany	○	1	≥100
4	PTSD in war regions	Burundi	●	5	≥2,200
5	PTSD in war regions	Uganda	○	1	≥200
6	Adverse childhood experiences	Germany	●	3	≥150
7	Learning deficits among medical students	Germany	●	3	≥200
8	Supporting parents after accidents of children	Switzerland	○	5	≥2,500
	Sum Σ			24	≥26,350

CN = Complex Navigation; PTSD = Posttraumatic Stress Disorder

tailored solutions. The questionnaires used in these scenarios (so-called *instruments*) not only provide questions, but also comprise sophisticated features for coordinating their processing (i.e., answering). For example, instruments require a proper navigation between questions based on already given answers. Recent approaches aim to realize such instruments as smart mobile applications to reduce the overall workload for domain experts by digitally transforming paper-based ways of data collection. For example, compared to traditional paper-based questionnaires, the data collected needs not to be digitized after completing an instrument, and, hence, mitigating transcription errors significantly.

To cope with these drawbacks, we propose a generic framework [9] that allows domain experts to create data collection instruments in a new way. According to this end-user programming approach, an instrument can be designed using a high-level modeling language (cf. Fig. 1, ①). The latter is then automatically transformed to an executable process model (cf. Fig. 1, ③), based on a well-defined mapping (cf. Fig. 1, ②). Finally, this process model can be deployed to mobile process engines running on smart mobile devices (cf. Fig. 1, ④).

Providing a process engine running as a *mobile service* on smart mobile devices, raises additional challenges. In particular, a modular architecture is indispensable. The following three major requirements must be particularly considered when developing such a mobile process engine:

R1 Provide offline execution. The mobile process engine shall allow for an offline execution of deployed process models as well as for storing the collected data on the smart mobile device. For example, in the Burundi project (cf. Table 1, #4), an international team of psychologists could not rely on robust Internet connection in rural areas.

R2 Enable process flexibility. The mobile process engine must support domain experts in changing (i.e., adapting) instruments during run time. For example, the order of questions often need to be flexibly changed in order to foster understandability of an instrument or to make it more convenient.

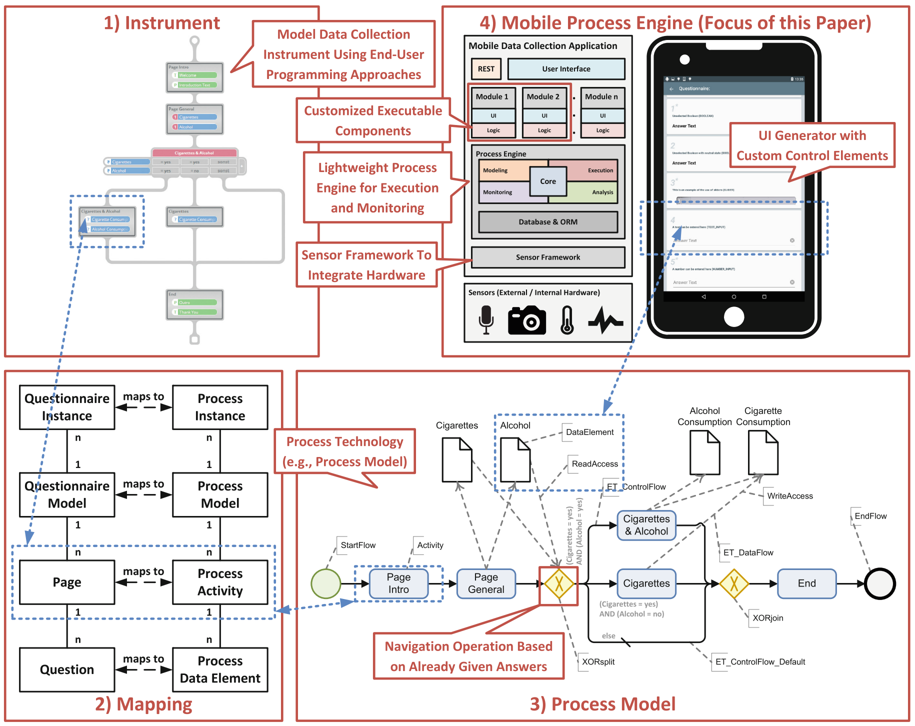

Fig. 1. The QuestionSys framework: (1) Modeling data collection instruments, (2) and (3) Mapping to process model, (4) Executing on smart mobile devices.

R3 Provide customizable user interfaces. The mobile process engine shall dynamically create the user interface of the respective instrument based on the model. For example, all information related to the structure and processing logic of the instrument as well as the meta-data of elements need to be taken into account by the rendering mechanism.

This paper presents a lightweight, modular mobile process engine, capable of executing sophisticated data collection instruments that takes multiple requirements into account. Moreover, a component for dynamically extending instruments is presented, which enables flexible adaptations of already deployed mobile applications during run time. Compared to hard-coded mobile data collection applications, therefore, changes of an instrument do not require its reimplementation and redeployment to multiple smart mobile devices. In addition, data from multiple releases must not be merged manually in order to avoid inconsistencies. Altogether, the approach enables flexibility regarding the design and execution of data collection instruments on smart mobile devices [8].

The remainder of the paper is structured as follows: Sect. 2 discusses fundamentals. Section 3 presents the architecture of the realized mobile engine, particularly its *Execution* component. Section 4 discusses related work and Sect. 5 concludes the paper.

2 Background: The QuestionSys Framework

This section introduces fundamentals of the QuestionSys framework, focusing on the mapping of a paper-based instrument to a mobile data collection application. Furthermore, the lifecycle phases for mobile data collection are introduced.

According to the QuestionSys approach, the structure of an instrument is directly mapped to an executable process model, which then can be enacted by a lightweight process engine running on smart mobile devices. Using this model-driven approach, a separation of the processing logic of an instrument from actual application code [7] of the data collection application becomes possible. Thereby, a process model acts as the schema for executing instrument instances. This model, in turn, consists of process steps (i.e., activities) and edges expressing the control and data flow between them. Additionally, gateways (e.g., AND and XOR-splits) provide functionality for describing complex control flow structures.

To properly support domain experts in creating a mobile data collection instrument, all phases of its lifecycle [9] shall be addressed. We introduce the lifecycle that consists of 5 different phases. The *Design & Modeling* phase enables domain experts to create sophisticated mobile data collection applications with complex logic themselves (i.e., end-user programming). The *Deployment* phase deploys the instrument on smart mobile devices. During the *Enactment & Execution* phase, multiple instances of the respective data collection instrument may be created and executed in a robust manner on the smart mobile devices. The *Monitoring & Analysis* phase provides functions enabling a real-time analysis of the data collected on the smart mobile device. Finally, the *Archiving & Versioning* phase enables release management for mobile data collection instruments.

The work presented in this paper focuses on the *Enactment & Execution* phase. In this context, a mobile service providing a lightweight process engine for executing data collection instruments has been developed.

3 QuestionSys Mobile Service

This section presents the architecture of the mobile process engine and provides in-depth information with respect to the *Execution* component.

The lightweight mobile process engine developed applies a service-driven approach. The engine comprises five components (cf. Fig. 2, left part): The most important one constitutes the core of the engine itself, which provides the data model, representing the process model, as well as operations to robustly interact with process instances (e.g., start or stop activities). Although, the process model relies on the ADEPT2 framework [6], other process meta-models (e.g., WS-BPEL) may be used as well. For this purpose, the core provides functions to import process models and map one model to another. The other components provide functions to support the different phases [10] of enacting process models locally on a smart mobile device. Note that these components only interact with the core itself and may be used as standalone functions as well (i.e., not all components are required). For example, the *Monitoring* component relies on

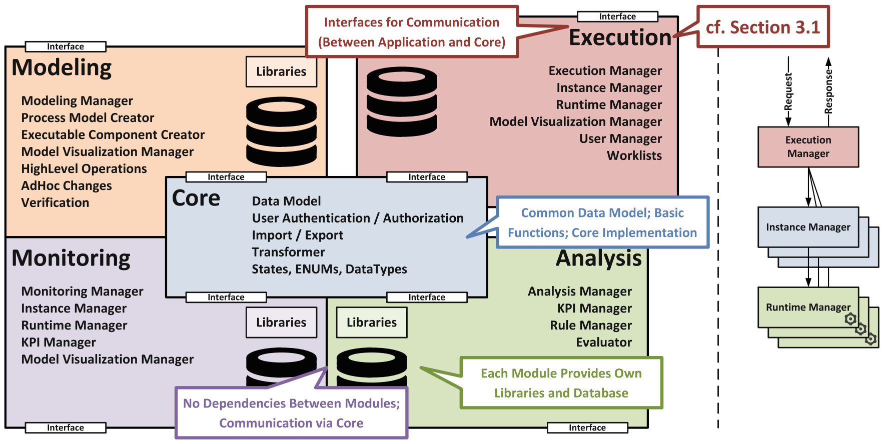

Fig. 2. Components of the mobile process engine

data from the *Execution* component in order to visualize the current state of the process instance or to provide information about upcoming process activities (e.g., insufficient data). This loose coupling of the components (e.g., no other dependencies between components exist) allows for a very customizable, but still lightweight mobile process engine.

As shown in Fig. 2, several components may provide similar functions. Consider, for example, the `ModelVisualizationManager` provided by the *Modeling*, *Execution* and *Monitoring* component. In general, these components require different functions of the respective managers (e.g., various notations) and, therefore, must be implemented several times. For example, the *Modeling* component needs to provide all elements of the process meta-model (e.g., process activities, data elements, control and data flow), whereas the *Execution* component may only provide information regarding the current and upcoming activities to be executed. The interface shared for this manager, however, is defined by the core of the mobile process engine. In addition, each component contains its own persistence layer. For example, the *Execution* component stores information about the current state of the enacted process instance (including user information, timestamps, data produced and consumed), whereas the *Analysis* component stores evaluation rules as well as the respective results for each process instance. These separated databases, in turn, foster the modular design of the process engine. Data between components, however, is shared through the core. Furthermore, each component may provide additional libraries to enhance functionality. For example, the *Analysis* component uses the Java Expression Language (JEXL) for evaluating data elements of process instances dynamically.

3.1 QuestionSys Mobile Execution Component

Recall that the mobile process engine runs as a service and may be embedded in another application based on its interfaces. The interaction between the mobile data collection application and the mobile process engine is shown in Fig. 3.

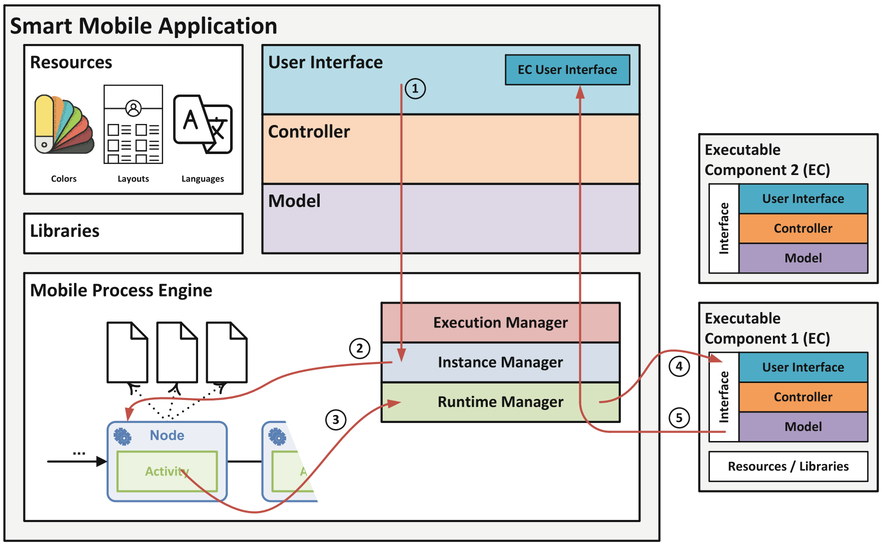

Fig. 3. Enacting executable components (`ECs`) during run time

First, the user, interacting with the smart mobile application, starts a new instance of an instrument. The mobile data collection application, in turn, directly interacts with the lightweight process engine, which provides access to the `ExecutionManager` ①. The latter offers functions allowing to control a particular process instance (i.e., move to the next page). *Second*, the `InstanceManager` validates whether the current node may be executed (e.g., the user has appropriate access rights) and all data elements needed are provided ②. The node is activated and handed over by a *third* step to the respective `RuntimeManager`, which is able to call the linked *executable component* (`EC`). The latter covers several aspects. The core functionality is to extract the main class file of the implementation of the `EC` as well as to create a list of all required *input* and *output variables* for the called component ③. *Fourth*, the `RuntimeManager` calls the `EC` by invoking its main method ④ and passing both input and output lists. As an `EC` can be seen as Micro Service [3], it may provide sophisticated logic as well as an user interface for interaction. Note that the `EC` may contain its own resource files as well as libraries. In a *fifth* step, the `EC` user interfaces are passed back to the `ExecutionManager` and the respective data collection application. This allows the latter to embed it as UI fragment inside the main user interface ⑤. Note that interactions with the UI of the `EC` (e.g., clicking a button) are handled by the `EC` itself and not by the *surrounding* mobile data collection application.

If the respective `EC`, which is executed as a mobile service, satisfies specific conditions (i.e., all mandatory fields are filled in), it produces the `canBeFinished` event. The latter indicates that the coordinating `RuntimeManager` may terminate the `EC`. Furthermore, all *output variables* of the `EC` are transferred back to the

`InstanceManager`, which stores them in respective *data elements* of the process instance. Log files collected during the execution of a specific instrument instance may be accessed by other components using the `ExecutionManager`.

In order to validate the presented architecture, a mobile application supporting researchers in collecting their data was realized. Altogether, the process-driven modeling supports researchers to easily create mobile data collection instruments. Furthermore, process technology enables the flexible execution locally on smart mobile devices in order to cope with domain-specific requirements.

4 Related Work

Executing business processes on mobile devices has been addressed by several approaches. Some of them provide proprietary execution languages specifically designed for respective scenarios, whereas others provide middlewares or frameworks enabling the development of process-aware mobile applications.

[1] presents extensions for WS-BPEL when integrating mobile devices into business processes. The authors discuss that in given scenarios the number of available mobile devices to coordinate is unknown. *Partner links*, bound to multiple endpoints, are introduced to cope with this issue.

In [5], an iPad application supporting medical staff during ward rounds is presented. Besides reviewing patient's health record or current diagnose, the staff is able to add further information during rounds. In order to execute a process, a lightweight process engine was implemented. Although the concept of automatically invoking processes based on user data is promising, the functionality of the respective engine is limited, as gateways are not supported, but only sequences of activities. Besides this limitation, only simple tasks may be executed.

[2,4] introduced a workflow engine being capable of running on PDAs. Both approaches use WS-BPEL to specify the business processes. Furthermore, they rely on Web Service standards (e.g., WSDL and SOAP) to specify activities to be called. Both use HTML for displaying user interfaces. In order to execute specific activities, one uses an own WS-BPEL extension, whereas the other ships with an Apache server to execute scripts. Both approaches provide core activities, like a browser, user forms, calendars and messaging services.

5 Summary and Outlook

Based on the insights we gained in several data collection scenarios, this paper advocates the need for sophisticated mobile services running on smart mobile devices. In order to mitigate the efforts between IT and domain experts, a sophisticated framework allowing domain experts to model data collection instruments themselves was proposed. In this context, a mobile service became necessary to process instruments directly on smart mobile devices. For this purpose, we present a flexible and modular architecture of a lightweight process engine. It allows extending the functionality of already installed mobile data collection applications during run time based on the concept of ECs. These components allow providing domain-specific logic as well as dynamically generated user interfaces for respective activities executed by the process engine.

To further validate the presented approach, a study for evaluating the user interface and user experience while working with the realized mobile data collection application is currently conducted. In particular, differences compared to paper-based questionnaires with respect to complex navigation features are evaluated. Moreover, the *Modeling*, *Analysis* and *Monitoring* components need to be implemented, leveraging the overall functionality of the proposed lightweight mobile process engine. In addition, ECs using sensors need to be realized allowing to collect additional data during enactment.

Altogether, the presented approach will significantly change the way instruments may be used in practice (e.g., clinical trials). Moreover, due to its flexibility, the proposed architecture may be suitable for other life domains relying on collecting and processing data in mobile scenarios as well.

References

1. Hackmann, G., Gill, C., Roman, G.C.: Extending BPEL for interoperable pervasive computing. In: IEEE International Conference on Pervasive Services, pp. 204–213. IEEE (2007)
2. Hackmann, G., Haitjema, M., Gill, C., Roman, G.-C.: Sliver: a BPEL workflow process execution engine for mobile devices. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 503–508. Springer, Heidelberg (2006). doi:10.1007/11948148_47
3. Newman, S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Inc., Sebastopol (2015)
4. Pajunen, L., Chande, S.: Developing workflow engine for mobile devices. In: 11th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2007, pp. 279–279. IEEE (2007)
5. Pryss, R., Mundbrod, N., Langer, D., Reichert, M.: Supporting medical ward rounds through mobile task and process management. Inf. Syst. e-Bus. Manag. **13**(1), 107–146 (2015)
6. Reichert, M., Dadam, P.: Enabling adaptive process-aware information systems with ADEPT2. In: Cardoso, J., van der Aalst, W. (eds.) Handbook of Research on Business Process Modeling. Information Science Reference, Hershey (2009)
7. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Heidelberg (2012)
8. Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Towards flexible mobile data collection in healthcare. In: 29th IEEE International Symposium on Computer-Based Medical Systems. IEEE Computer Society Press, June 2016
9. Schobel, J., Pryss, R., Schickler, M., Reichert, M., Ruf-Leuschner, M., Elbert, T.: End-user programming of mobile services: empowering domain experts to implement mobile data collection applications. In: IEEE 5th International Conference on Mobile Services. IEEE Computer Society Press, June 2016
10. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer Science & Business Media, Heidelberg (2012)