Context-based Handling of Mobile Process Activities

Rüdiger Pryss and Manfred Reichert
University of Ulm, Institute of Databases and Information Systems, Germany

ABSTRACT
Process technology constitutes a crucial component of information systems. In this context, high flexibility is required as business functions must be quickly adaptable to cope with dynamic business changes. As recent developments allow for the use of mobile devices in knowledge-intensive areas, it is often demanded to enhance process-aware information systems with mobile activity support. In general, the technical integration of this activity type with existing process management technology is challenging. For example, protocols governing the communication between mobile devices and process management systems must be adapted. If a mobile context shall be additionally considered, the integration gets even more complex. However, the use of a mobile context offers advantages. For example, the mobile activity execution time may be decreased if mobile activities are only assigned to those users whose location is beneficial. This work proposes an approach to enable the robust handling of single process activities on mobile devices based on a mobile process model.

Keywords: Mobile Service, Human-centric Mobile Activities, Mobile Context, Mobile Process Model, Mobile Worklist, Mobile User Assignment, Exception Handling, Exception Prevention

INTRODUCTION
Daily business routines more and more require mobile access to Information Systems. However, the integration of mobile devices into existing infrastructures is laborious and error-prone. In particular, the infrastructure must cope with ad hoc events, various types of exceptions (e.g., connectivity problems), physical limitations of mobile devices (e.g., limited battery capacity), misbehavior of users (e.g., instant shutdowns), and the evaluation of data collected by mobile sensors (Schobel et al., 2013). In general, proper exception handling constitutes a prerequisite for any mobile activity support. In this context, adaptive and flexible process management technology offers promising perspectives based on a wide range of techniques (Reichert & Weber, 2012; Reichert & Weber, 2013). In particular, it allows for the proper handling of run time exceptions. However, execution of process activities on mobile devices in the same way as on stationary computers is not appropriate when the specific challenges of mobile environments are not taken into account.

A service-oriented environment should allow for mobile activity support during business process execution. This paper presents an approach developed in the MARPLE (Managing Robust Mobile Processes in a Complex World) project. This approach enables the robust execution of single process activities on mobile devices and is based on two services, a service that assigns mobile users to mobile activities and an exception handling service for mobile activities. These services ensure that mobile activities are (a) only assigned to those mobile users that are particularly appropriate based on a mobile context and (b) do not harm the overall process execution when activity exceptions occur. In this context, a service-oriented architecture was realized that integrates the services with existing process management technology. To be more precise, the architecture allows for the instantiation, activation, and exception handling of mobile activities.

This paper presents the support of *mobile activities* and the handling of exceptions during run time without need for manually involving mobile users. Note that this is crucial with respect to higher user acceptance of mobile business processes. Generally, the provisioning of self-healing techniques is crucial for executing mobile activities in the large scale as well as for achieving higher user acceptance.

We firstly discuss fundamental issues arising in the context of mobile environments. Their understanding is crucial for developing the two fundamental services as well as for designing the overall system architecture. In this context, the challenges (e.g., device failures) are considered which must be tackled to ensure robust execution of mobile activities. In detail, challenges are addressed that are related to the mobile environment itself (e.g., a mobile device loses its connectivity), related to the business process execution (e.g., missing data caused by activity exceptions), and related to the behavior of the mobile users (e.g., instant shutdowns).

BACKGROUND

Many domains crave for the integration of mobile devices into business process execution (Lenz & Reichert, 2007; Pryss et al., 2016(a)). Figure 1 shows a simplified healthcare example illustrating this. It depicts a ward round process for which mobile assistance is required (Pryss et al., 2015). For instance, *Prepare Ward Round* constitutes an activity whose mobile support would ease daily work of healthcare professionals.

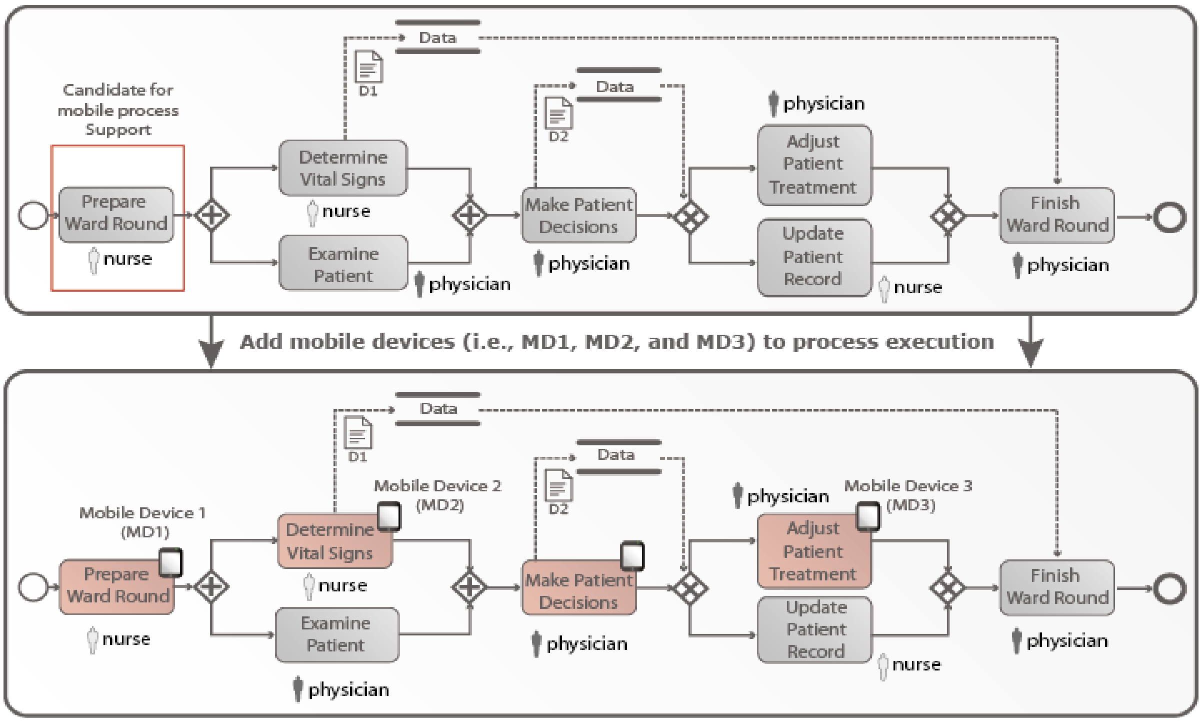

Figure 1. Adding mobile devices to process execution (mobile activities are indicated with an icon)

The use of mobile devices during process execution raises several challenges with respect to mobile activity support. For example, if the mobile device running the activity *Determine Vital Signs* (see Figure 1) encounters physical problems, overall process execution might be harmed; or if activities succeeding a mobile activity in the flow of control have to access data that is usually provided by this mobile activity, standard exception handling strategies (e.g., to skip the mobile activity) are not appropriate when the mobile activity fails. As shown in Figure 1, activity *Finish Ward Round* is data-dependent on mobile activity *Determine Vital Signs*. In turn, this might cause problems when activity exceptions occur, i.e., if *Determine Vital Signs* fails, the process cannot terminate properly, since activity *Finish Ward Round* cannot be properly executed due to the missing value of data element *D1*.

Challenges for Executing Processes with Mobile Activities

To be able to run selected activities on mobile devices during process execution, the challenges imposed by mobile environments need to be properly addressed. These challenges relate to the state of mobile devices as well as the behavior of mobile users. In addition, the specific challenges relating to process execution must be considered as well; e.g., dealing with missing process data due to failed mobile activities at run time. This section presents backgrounds on the elicitation of challenges in this context, discusses relevant and challenges addressed, and categorizes them into process-, mobile environment- and user- related.

In the following, three advanced mobile application scenarios will be discussed. These scenarios (case studies) base the considerations on relevant challenges when executing mobile activities in the context of business processes. Thereby, only one scenario is relevant in the context of business process execution. The remaining two scenarios are not concerned with business process execution, but also constitute demanding scenarios for mobile application development in general. As a result, they revealed similar challenges as have been identified for mobile activity execution. The first mobile application scenario stems from the healthcare domain, whereas the second from the field of augmented reality, and the last one from the psychological domain.

Supporting Medical Ward Rounds with Mobile Task and Process Management
In hospitals, ward rounds are crucial for decision-making in the context of patient treatment processes. In the course of a ward round, new activities are defined and allocated to healthcare professionals. In clinical practice, however, these activities are not properly addressed. During ward rounds, they are jotted down using pen and paper, and their later processing is prone to errors. Furthermore, healthcare professionals must keep track of the processing status of their activities (e.g., medical orders). To relieve healthcare professionals from such a manual activity management, the MEDo (MedicalDo) approach (Pryss et al., 2015) supports ward rounds by transforming the pen and paper worksheet to a mobile user interface on a tablet integrating process support, mobile activity management, and access to the electronic patient record. Based on the lessons learned from this case study, requirements have been derived for mobile activity support in the context of medical ward rounds. As a particular challenge, the following has been identified: if mobile assisted activities will be interrupted and not properly continued later, exceptions like missing data frequently occur. Altogether, Table 1 summarizes which challenges in terms of parameters for executing processes with mobile activities have been identified from this mobile application scenario.

Location-based Mobile Augmented Reality Applications
In the AREA (Augmented Reality Engine Application) project (Pryss et al., 2016(a); Geiger et al., 2013), an advanced mobile application, which enables location-based mobile augmented reality on three different mobile operating systems (i.e., iOS, Android, and Windows Phone) has been designed and implemented. This kind of mobile application is characterized by high resource demands since various sensors must be queried at run time and numerous virtual objects may have to be drawn in real time on the screen of the mobile device. Therefore, the lessons learned when implementing real world mobile business applications with the mobile augmented reality engine, have revealed profound insights into challenges of mobile application development in general. Table 1 summarizes again which challenges in terms of parameters for executing processes with mobile activities have been identified from this mobile application scenario.

Applying Mobile Technology to Psychological Questionnaires
Many psychological studies are performed with specifically tailored "paper & pencil"-questionnaires. Such a paper-based approach usually results in a massive workload for evaluating and analyzing the collected data afterwards, e.g., to transfer data to electronic worksheets or any statistics software. To relieve researchers from such manual tasks and to improve the efficiency of data collection processes, mobile device applications have been

developed in the context of the QuestionSys[1] project for existing psychological questionnaires (e.g., Schobel et al., 2016(c)). Based on these applications, the usefulness of mobile devices for mobile data collection in the context of psychological questionnaires has been demonstrated. Although the implemented applications already have shown several advantages in respect to data collection and analysis, they have not been suitable for psychological studies in the large scale yet. Consequently, new challenges emerged when using the mobile applications in such demanding scenarios. Table 1 summarizes which challenges in terms of parameters for executing processes with mobile activities have been identified from this mobile application scenario as well.

Considered Context for Executing Processes with Mobile Activities

The mobile context is represented by a parameter catalogue. The parameters were identified in the discussed real-world scenarios. In addition, the catalogue is based on a comprehensive literature study. For several reasons, we assign parameters to four categories: First, parameters related to the mobile device of a mobile user (SMD parameters) are managed. For example, a battery status is managed for mobile devices. Second, all parameters associated with mobile activities (MA parameters) are managed. For example, the execution location of the mobile activity is captured. Third, parameters that can be related to the overall process execution are managed. This category became necessary to cope with the complexity of managing a multitude of parameters. For several parameters, it would be costly to manage them separately for each mobile activity. Therefore, a parameter applying to all mobile process activities is used, e.g., to manage a generally demanded battery status for the activation of all mobile activities of a process. Finally, parameters associated with mobile users (MU parameters) are managed, e.g., the usual location of a mobile user. The entire parameter catalogue is presented in Table 1. In addition, Table 1 presents in which real world scenarios a parameter was crucial. Column *T* of Table 1 further indicates whether a parameter is of type symbolic or measured. From the considered application scenarios, we revealed that such differentiation is useful. Symbolic parameters are used in related work to define parameters on an abstract level (Becker & Dürr, 2005). For example, regarding the location of a mobile activity, the symbolic parameter emergency room might be used. First, symbolic parameters are considered as they can already be evaluated before starting a process. For example, consider the following scenario: Symbolic parameters are managed for mobile users and mobile activities. Hence, it can be indicated before the execution of a mobile activity takes places how many mobile users are closely located to the location of the mobile activity that shall be performed based on a simple comparison of the symbolic parameters. Apparently, it is just an indication, but has shown its advantages in practice. Second, for the assignment of mobile users to mobile activities symbolic parameters can be advantageous as well. For example, if precise location information (e.g., GPS) cannot be obtained for a user location due to a connection loss, the symbolic parameter may be used instead. Conversely, measured parameters are automatically determined after starting the process instance. For example, if a mobile activity shall be executed, the battery status of all mobile users is gathered.

Selected parameters require a brief discussion. The form factor parameter is used to indicate the mobile device type (i.e., tablet or smartphone). Geometric coordinates, in turn, are measured and correspond to GPS coordinates in an outdoor scenario and WLAN coordinates in an indoor

[1] Further information to the project can be found at: http://www.dbis.info/questionsys

scenario. Parameter location range refers to the location of a mobile activity defining a radius around its geometric coordinates. Mobile users located inside this radius are considered for executing this mobile activity. Urgency value defines a period (or point in time) during which the mobile activity shall be executed. Furthermore, response frequency is a value that determines the frequency with which the mobile device of a particular user must report its online status to the process management system. If the mobile device does not obey this reporting frequency, an exception handling is triggered. Related to response frequency is parameter offline mode. If the latter is set to *true* for any mobile activity, the mobile device may ignore the response frequency to enable offline execution of this mobile activity. In practice, this was frequently demanded. Finally, network type is used to capture the network connection of a mobile device (e.g., WLAN or UMTS).

In addition, the parameter catalogue contains two threshold parameters. User threshold indicates the number of users that need to be available to activate a mobile activity. In turn, the instant shutdowns parameter captures the behavior of mobile users. Note that in practice users may instantly shutdown their mobile device without reflecting on the consequences of this shutdown. Usually, this constitutes a short-term problem and the device can be restarted soon in most cases. If a user exhibits many instant shutdowns, however, this misbehavior should be considered in the context of mobile activity assignments. To cope with such careless shutdowns, a mobile device sends a message to a service that an instant shutdown will take place soon. In this context, several mobile development frameworks were evaluated (i.e., Google Android, Apple iOS, Microsoft Windows Mobile) and it could be demonstrated that this solution for detecting instant shutdowns is feasible for all of them. Finally, to assess user behavior over time (e.g., whether or not a user performs many instant shutdowns), parameter instant shutdowns is managed. Related to it, a general process parameter instant shutdown threshold is managed for all mobile activities that is compared with the parameter managed for a particular mobile user. If the parameter value of a mobile user is above the instant shutdown threshold parameter, he or she will be particularly considered for the execution of the mobile activity.

Regarding the presented parameter catalogue, we do not claim that it captures all possible or required parameters. It rather reflects insights we gathered from our analysis of real-world scenarios. Furthermore, the application of the parameters identified in other practical scenarios was promising. More specifically, domain experts were able to determine useful parameter values.

	Parameter	***Description***	***Type***	***MEDo***	***AREA***	***QuestionSys***
Mobile Environment		Category I: Mobile device (SMD)				
	SMD_{BS}	Battery Status	M	x	x	x
	SMD_{FF}	Form Factor	S	x	x	-
	SMD_{NT}	Network Type	M	x	x	x
	SMD_{GC}	Geometric Coordinate	M	x	x	-
Process		Category II: Mobile Activity (MA)				
	MA_{SC}	Symbolic Coordinate(s)	S	x	-	-
	MA_{GC}	Geometric Coordinate	M	x	x	-

	MA_{LR}	Location Range	S	x	x	-
	MA_{BS}	Battery Status	M	x	x	x
	MA_{U}	Urgency Value	S	x	-	-
	MA_{OFF}	Offline Mode	S	x	x	x
	MA_{FF}	Form Factor	S	x	x	x
	MA_{RF}	Response Frequency	S	x	x	x
	MA_{UT}	User Threshold	S	x	-	-
	Category III: Process (P)					
	P_{IST}	Instant Shutdown Threshold	S	x	-	-
User Behavior	Category IV: Mobile User (MU)					
	MU_{SC}	Symbolic Coordinate(s)	S	x	-	-
	MU_{IS}	Instant Shutdowns	S	x	-	-
	| Type: M=Measured, S=Symbolic | (x): relevant | (-): not relevant |					

Table 1. Excerpt of considered real world projects revealing parameters for executing processes with mobile activities

Mobile Process Execution Approaches

Next, we will discuss alternatives for realizing processes that comprise mobile activities and describe corresponding process models. For realizing processes with mobile activities, three different approaches exist (see Figure 2):

Approach 1 *(Physical Process Fragmentation aka Process Partitioning):*

A process (i.e., process schema) is physically partitioned during design time. The resulting process fragments and their activities are then assigned to a number of mobile devices before run time (and on an instance-per-instance basis, see Approach 1 in Figure 2). Consequently, the execution of process fragments must be synchronized during run time. This is a complex task to accomplish. For example, if the same data element is written by different process fragments, sophisticated synchronization techniques become necessary to ensure data consistency. Another challenge emerges if a device encounters physical problems (e.g., a lost connection).

Approach 2 *(Logical Process Fragmentation aka Migrating Processes):*

A process schema is partitioned logically. In this case, the resulting process fragments and their activities are executed on different mobile devices. Contrary to the first approach, the original process schema will be preserved during run time when executing the process fragments (Approach 2 in Figure 2). Usually, migration techniques are applied in this context (Zaplata et al., 2010). More precisely, based on the original process schema, it can be determined how the migration between logical process fragments shall be accomplished at run time. Accordingly, it is dynamically determined which device shall execute which process fragments. In particular, this allows for dynamic exchanges of devices already assigned to a fragment. Another challenge emerging in this context is synchronization of the execution of parallel process branches concurrently executed on different mobile devices, i.e., a synchronization method is required to cope with data inconsistencies when joining the execution of the different branches.

Approach 3 *(Single Mobile Activity Handling):*
Single process activities are executed on mobile devices. For this purpose, a mobile device must cover a subset of the functionality of a stationary process client, e.g., a worklist component that is continuously updated by the process engine (see Approach 3 in Figure 2).

Discussing the Approaches and Related Work
In the following, only a brief discussion about Approaches 3 will be given. Approaches 1 and 2 are predominant in research on distributed processes in general and for distributed mobile processes in particular. However, the robust execution of mobile activities (i.e., Approach *3*) has not been researched extensively. As this paper deals with *Approach 3*, only this approach is discussed in more detail. More information to the other approaches can be found in (Pryss & Reichert, 2016). In general, by following Approach 3, our focus is on the robust execution of mobile activities*,* while at the same time not burdening mobile users when exceptions occur. Note that *Approach 1* is generally not considered in MARPLE as it reveals too many drawbacks. Approach 2 is considered in MARPLE, but not subject to this paper. Figure 2 finally summarizes all presented approaches.

Approach 3 and related work: Note that Approach 3 only focuses on single mobile activities of a business process. Therefore, only related work is important which focuses on the challenges to perform single activities on mobile devices properly. In (Alonso et al., 1995), the challenges of disconnected clients in the context of business process execution have been early discussed with no mobile context. Since disconnections of a device performing an activity constitute the most important aspects for mobile activity execution, (Alonso et al., 1995) can be regarded as first work dealing with challenges relevant for mobile activities in the context of business processes. In sequel to that early research work, less work (e.g., (Tuysuz et al., 2013)) on how to properly execute mobile activities on mobile devices exist. In particular, many characteristic challenges imposed by mobile environments are less considered. Only the aspect of missing or inconsistent data is predominant subject to many research works in this context (Hahn & Schweppe, 2009). For example, (Hahn & Schweppe, 2009) proposed the apply transaction techniques to mobile processes. To deal with failures of mobile devices (e.g., a disconnected device) mobile activities are executed within transactions. Furthermore, transactional properties are defined to determine in what cases a transaction has to be cancelled. Used techniques in this and similar work do not deal with exceptions like presented in this paper. In particular, they do not focus on how to compensate exceptions while ensuring same execution semantics as originally intended. Another research work relevant in this context is proposed by (Pryss et al., 2015). The presented MEDo approach deals with mobile task handling in the context of medical ward rounds. In turn, MEDo does not deal with fine-grained considerations on mobile activities, i.e., user-, process-, and environment challenges, as constituted by the above presented seven challenges.

Furthermore, recent related works address exception handling in the context of mobile services (Chen, Cardozo, & Clarke, 2016; Marinescu, et al., 2015). However, they do not focus on human-centric mobile activities. Furthermore, preventing exceptions in a mobile context is not explicitly considered. For example, approaches dealing with mobile agents (Marinescu, et al., 2015) focus on exception prevention. Again, they do not specifically consider human-centric mobile services as the presented framework does. Finally, existing commercial process management systems supporting the integration of mobile activities do also not provide a particular exception handling concept (IBM, 2018).

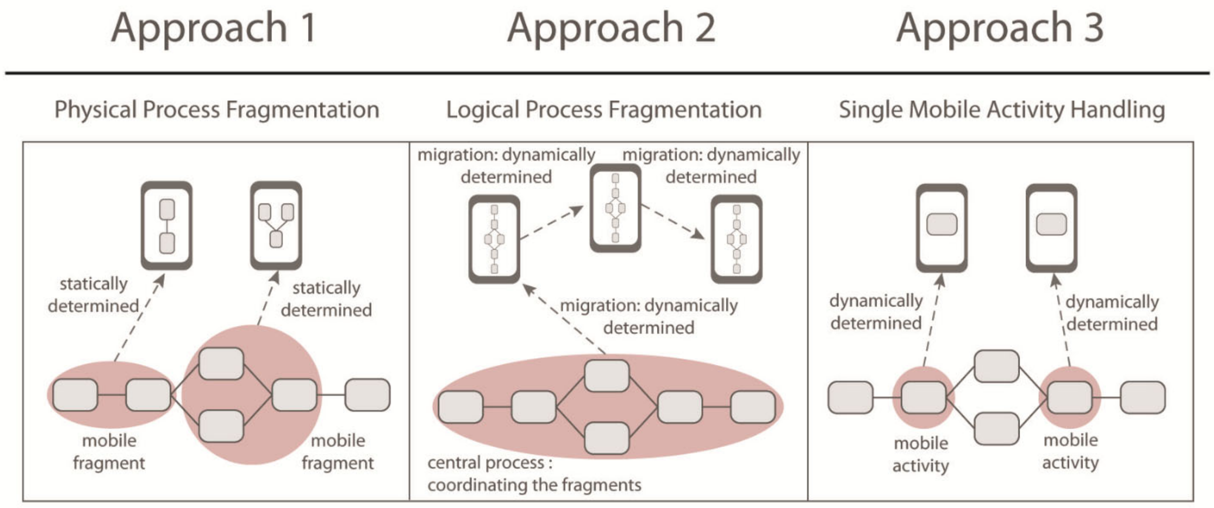
Approach 1
Approach 2
Approach 3
Physical Process Fragmentation
Logical Process Fragmentation
Single Mobile Activity Handling
statically determined
statically determined
mobile fragment
mobile fragment
migration: dynamically determined
migration: dynamically determined
migration: dynamically determined
central process : coordinating the fragments
dynamically determined
dynamically determined
mobile activity
mobile activity

Figure 2. Approaches for realizing mobile processes

Defining Processes with Mobile Activities

This section introduces the notion of our developed process meta-model that is crucial for the match-making model (i.e., assigning mobile activities to concrete mobile users) of the mobile activities (see Fig. 3, 1). The meta-model, in turn, is based on an extensive literature review (e.g., (Zaplata et al., 2010; WFMC, 2018)). Consider that Fig. 3 also illustrates mobile worklist management in the context of the meta-model (see Fig. 3, 6; (Pryss et al., 2016(b))). Note that all entities concerned with worklist management are marked accordingly. As can be seen, fundamental adaptations were required to enable worklist management for mobile activities. The meta-model is denoted as *mobile process meta-model*. In the context of worklist changes, two algorithms were developed (see Fig. 3, 2&3). The first algorithm manages user assignments and the execution of mobile activities (see Fig. 3, 2). It is denoted as *Selection Algorithm*, meaning that all mobile users determined by the algorithm are qualified to execute the mobile activity. The qualification, in turn, is based on three concepts: authorization (e.g., roles (Sandhu et al., 1996)), constraints; e.g., to ensure two activities will be executed by the same user (Pryss, Musiol, & Reichert, 2013), and a mobile context (see Fig. 3, 4). As frequently changing circumstances and limited resources have to be properly addressed in mobile environments, the goal should be to find those mobile users with appropriate capacities (e.g., being closely located to the place the mobile activity shall be enacted) on one hand. On the other, the determined mobile users shall minimize the occurrence of exceptions. In this context, the *Selection Algorithm* evaluates the resource situation of all qualified mobile users as well as the best environment matching (e.g., the user is closely located to the mobile activity.). Note that a good matching in these aspects revealed quicker execution times of mobile activities and hence reduces exceptions. The second algorithm, the *Ranking Algorithm*, handles exceptions (e.g., mobile device crashes) during the execution of mobile activities (see Fig. 3, 3). To enable a proper exception handling, changes in respect to the state model of mobile activities (see Fig. 3, 5) became necessary, i.e., compared to non-mobile activities, the behavior of specific state transitions had to be changed and new states had to be added.

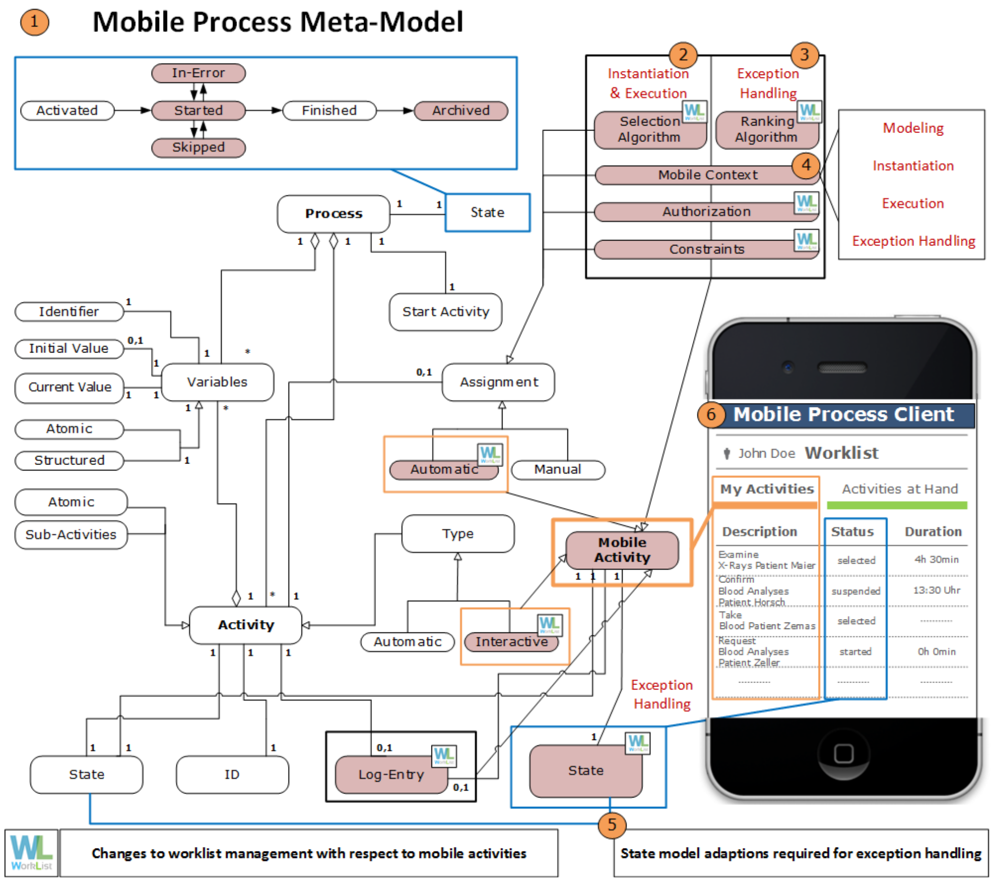

Figure 3. Mobile Process Meta-Model

MOBILE PROCESS ACTIVITY INTEGRATION ARCHITECTURE

This section presents the proposed architecture in terms of the delegation and backup concept and discusses the management of user lists for the delegation service. First of all, the run time architecture is sketched, in which delegation and backup services are applied.

Mobile Activity Run Time Architecture

The delegation and backup services allow for the robust execution of mobile activities. Furthermore, their design allows for the use of existing service-driven business process engines. Implementing a specific process engine which provides all functions for creating and executing mobile activities constitutes another possible direction. However, if a process management system is already in use, the introduction of a new process engine is usually a complex endeavor due to high efforts for transferring process models to the new engine. Therefore, the presented architecture provides an engine-independent interface for executing mobile activities. Since communication with Web services constitutes a core feature of any modern process engine, a service-driven approach has been realized (see Figure 4). The core of the run time architecture for executing mobile activities is denoted as mobile execution environment (see Figure 4). Note that the mobile execution environment extends existing business process environments without any mobile support. It provides components to manage mobile users (list management), to handle delegations, to perform the backup service, and the Interfaces 1 and 2. The list management component, in turn, maintains user lists to enable delegation management. Interface 1 is the connection to a business process management environment. Its purpose is to communicate with the business process engine. It is designed and implemented to allow for the integration of a wide range of existing business process environments (e.g., Intalio (Ghalimi, 2006) or Activiti (Meister, 2011)). Finally, Interface 2 is used to integrate mobile devices with the mobile activity execution. Its purpose is to communicate with the mobile devices. Furthermore, a mobile service client has been implemented which is deployed to mobile devices. This client is used to push the following information via Interface 2 to the mobile execution environment: battery status, connection status, user location, and instant shutdowns. Thereby, the mobile device periodically sends status of user location, connectivity, and battery charge level to the mobile execution environment. Note that instant shutdowns are determined as follows: if a mobile device gets offline and recovers, it will be determined whether an instant shutdown has been the reason for getting offline. For example, regarding Android devices, the "<action android:name="android.intent.action.ACTION_SHUTDOWN"/>" will be determined. If an instant shutdown has been identified, the shutdown counter for the respective device will be increased.

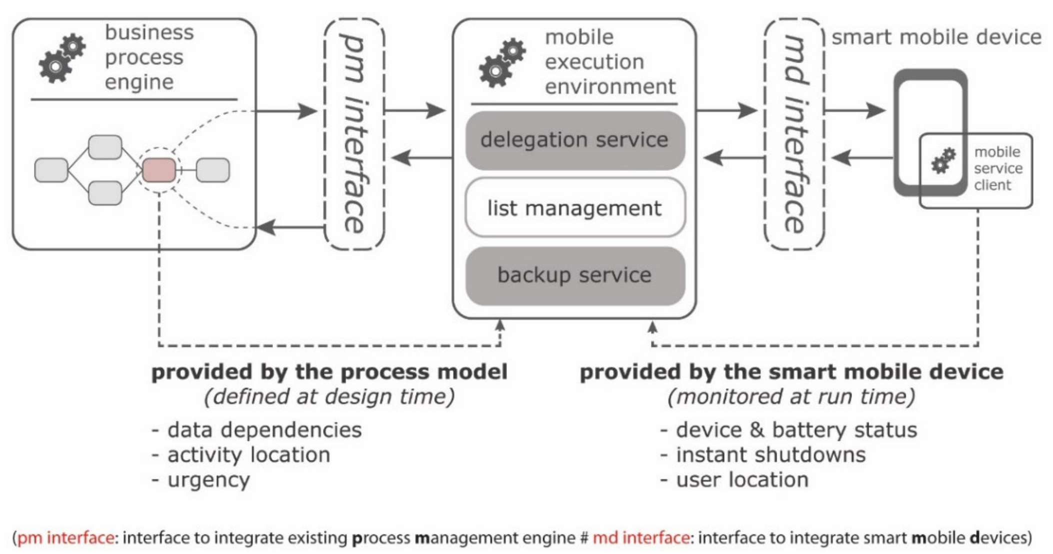

Figure 4. Mobile activity run time architecture (simplified)

Delegation

We introduce the notion of *delegation* during process execution as well as a corresponding approach (see Figure 5).

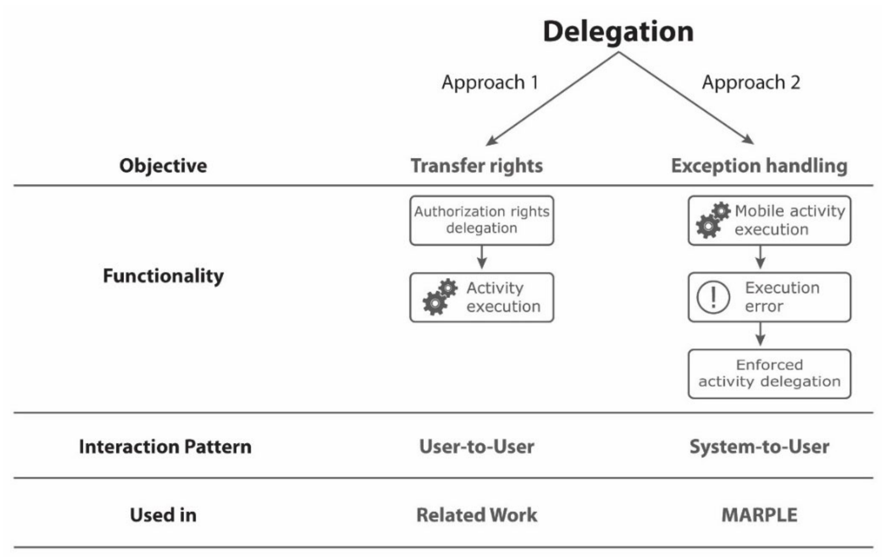

Figure 5. Delegation mechanisms

***Approach 1** (Transfer Rights)*:
Delegation permits a user to assign all or a subset of their authorization rights to other users not possessing these rights at the moment (see the *functionality of Approach 1* in Figure 5). Thereby, they may either assign these rights in the context of a particular process instance or for all process instances. (Schaad, 2003; Crampton & Khambhammettu, 2008; Gaaloul & Charoy, 2009) provide various reasons for delegating respective rights; e.g., a user may not possess required documents, or an entailment constraint like separation of duties must be enforced (Pryss, Musiol, & Reichert, 2013). In this context, delegation may be accomplished based on two techniques: First, delegation may be applied based on user-defined rules stored in a repository. Second, users may delegate their rights dynamically during run time. The two techniques can be summarized as *user-to-user* interaction pattern: A user determines the context in which a delegation may be applied. Finally, delegations only take place when an activity is in a desired state. Note that *Approach 1* constitutes the common notion for delegation in the context of business processes.

***Approach 2** (Exception Handling)*:
Delegation may be further applied in the context of exception handling. It will then be performed when mobile users and their devices encounter problems. This pattern is applied in the proposed architecture. Note that a delegation may be only performed to another mobile user possessing the same rights as the mobile user who has caused the problem. The delegation will then be performed in a *system-to-user* interaction pattern.

User List Management for Delegation and Backup

To foster robust execution of mobile activities, three different user lists are maintained: an initial user list ul_{init}, a user list ul_{mob} comprising appropriate mobile users, and a delegation list dl_{mob}. Note that all lists except ul_{init} are maintained by the mobile execution environment. ul_{init} is provided by a process engine and contains all mobile users u_{mob} authorized to perform a mobile activity t_{mob}. Furthermore, ul_{init} constitutes the basis for determining the two other lists, which are determined based on an analysis of ul_{init}. Thereby, ul_{init} is created by considering the following properties:

Property	***Description***
Connectivity	Indicates whether a user u_{mob} is online or offline.
Low Battery Status	Indicates whether the user's device has a low battery status.
User Location vs. Activity Location	The user's current location will be compared to the activity location. For example, if the *location* attribute of a mobile activity has value *emergency unit* and a mobile user is currently staying at another ward, he/she will not be considered for ul_{init}. Moreover, during this phase both the symbolic as well as the geometric coordinates are used (see Table 1)
Pre-filter	Indicates if u_{mob} has been excluded by a *pre-filter.*
Instant Shutdowns	The user's current number of instant shutdowns will be compared to the generally defined threshold for instant shutdowns of a mobile activity.

Table 2. Properties for determining ul_{init}

As soon as mobile activity t_{mob} becomes activated, ul_{mob} will be calculated by the *delegation service* as follows (see Table 3):

```
ul_mob ← {}
FOREACH u_mob IN ul_init
        IF (u_mob.connectivity) AND (¬ u_mob.pre-filter)
                IF (u_mob.location = t_mob.location) or (u_mob.location = {} and t_mob.location=0)
                        THEN
                                ul_mob.append(u_mob)
FOREACH u_mob IN ul_mob
    u_mob.determine.ranking.value(u_mob.resource.behavior¹)
ul_mob.sort() based on ranking.value and in descending order
```

¹= *resource.behavior is a complex procedure not shown here due to space limitations. It evaluates the resource behaviour of a mobile user (based on connection losses, performed instant shutdowns, failed delegations, and low battery times)*

Table 3. Ranking Algorithm

According to this procedure, all mobile users from ul_{init} being online, not being pre-filtered (i.e., not being manually excluded for the process or an instance of the process), and are located closely to the required location of the activity (again, both the symbolic and geometric coordinates are evaluated), will be appended to ul_{mob}. Then, ul_{mob} will be sorted in descending order based on a ranking value. The ranking value is determined for each mobile user and determines his resource behaviour. The latter is calculated based on the connection losses, performed instant shutdowns, failed delegations, and low battery times of a mobile user. Thereby, a ranking with high value indicates good resource behaviour for a mobile user; i.e., he has exhibited less connection losses, performed less instant shutdowns, caused less failed delegations, and showed less times a low battery status than other mobile users with a lower ranking value. Finally, the mobile user in ul_{mob} with highest ranking value will be notified about the mobile activity being ready for execution. As a result, he will see this activity in his work list on the mobile device. After the mobile user has claimed the activity, the delegation can be started. If he declines the execution, ul_{mob} will be used to identifiy the mobile user next in line with respect to the ranking value. Then, this identified mobile user will be notified about the mobile activity being ready for execution. The procedure is repeated until a mobile user accepts the delegation request or he constitutes the one in ul_{mob} with the lowest ranking value (i.e., the final mobile user in ul_{mob}). If the latter is the case, the mobile user cannot decline the delegation request.

User Assignment and Race Conditions
The different lists maintained and presented above for delegation and backup service prevent race conditions with respect to user assignments. As a result, only one mobile is responsible for performing a mobile activity at any point in time. This will be enabled due to the following reasons:

- Delegation ensures that a mobile user performing a mobile activity is distracted from the respective mobile activity execution before delegation to another user is performed.
- User list management ensures that work lists on the mobile devices are synchronized and only one user can claim a mobile activity.
- Delegation management only prioritizes mobile users according to the presented aspects (e.g., battery status). As a result, the assignment of users is an atomic operation since it is performed similar to user list management in existing business process engines (e.g., Reichert et al., 2009).

Protocol Management for Delegation

In a process management system that realizes that mobile process meta-model we developed, the delegation concept needs a protocol definition between the mobile device and the process management system. Therefore, the protocol coordinating the interactions between the mobile process client and the process management system is presented (see Fig. 6). First, we present required components (see Fig. 6), which were implemented as a service-oriented middleware that interacts between an existing process management system and the mobile process client. The basic components of the middleware include a service-centric play application and a MySQL database connected to it. The delegation concept presented in this work is realized by the *mobile activity handler*. Processes, in turn, are coordinated by the adaptive process management system AristaFlow BPM Suite. The realized mobile process client consists of two components that manage the entire communication: the worklist client and the execution client. Thereby, the worklist client manages the worklist, whereas the execution client manages the communication between the worklist client, an invoked mobile application, and the developed service-oriented middleware. The invoked mobile applications, in turn, actually perform the mobile activity (e.g., invoking Mobile Microsoft Excel). Based on this, the delegation protocol was realized. It governs the interactions between the mobile process client and the service-oriented middleware in case of a delegation. The protocol steps are depicted in Fig. 6. Thereby, steps within the *In-Delegation* box are crucial for handling delegations. In particular, they constitute the steps performed after a delegation. Two scenarios must be distinguished. First, the mobile device might no longer work after the occurrence of the exception. Second, it might still work, but no longer be connected to the process management system. In the first case, all steps shown for the mobile process client are not performed. In the second case, all shown steps are performed. After starting the delegation, the process management system performs the following steps. First, it withdraws the mobile activity running on the mobile device by updating its status (see Steps 10'-11'). Second, after updating the status it determines whether the mobile device has reconnected in case the connection loss was only a short-term problem (see Step 12'). Third, depending on the result of Step 12', it may start the delegation. The mobile process client, in turn, applies the following steps. First, the running activity is stopped and the data created is locally cached (see Steps 10-13). Second, after the mobile device reconnects to the process management system, it requests the status of the delegation (see Step 14) and sends its cached data to the process management system. Two additional scenarios need to be distinguished (see Fig. 6, 6) after a reconnection. First, if a delegation has not been accomplished yet, the reconnecting mobile device gets the activity execution back. Second, if the delegation is still running, the cached data is transferred to the mobile device currently performing the delegation. This way it can be ensured that no data is lost. In addition, a feature was realized that enables recipients to manually decide whether or not to use cached data before it will be actually transferred. Furthermore, we identified the protocol points at which the mobile context parameters shall be exchanged between the mobile process client and the service-oriented middleware. Fig. 6 (2,4) depicts two protocol points at which the service-oriented middleware requests parameter values from the mobile process client. In turn, Fig. 6 (1,3,5) shows the points in time at which the mobile process client sends parameter values to the service-oriented middleware.

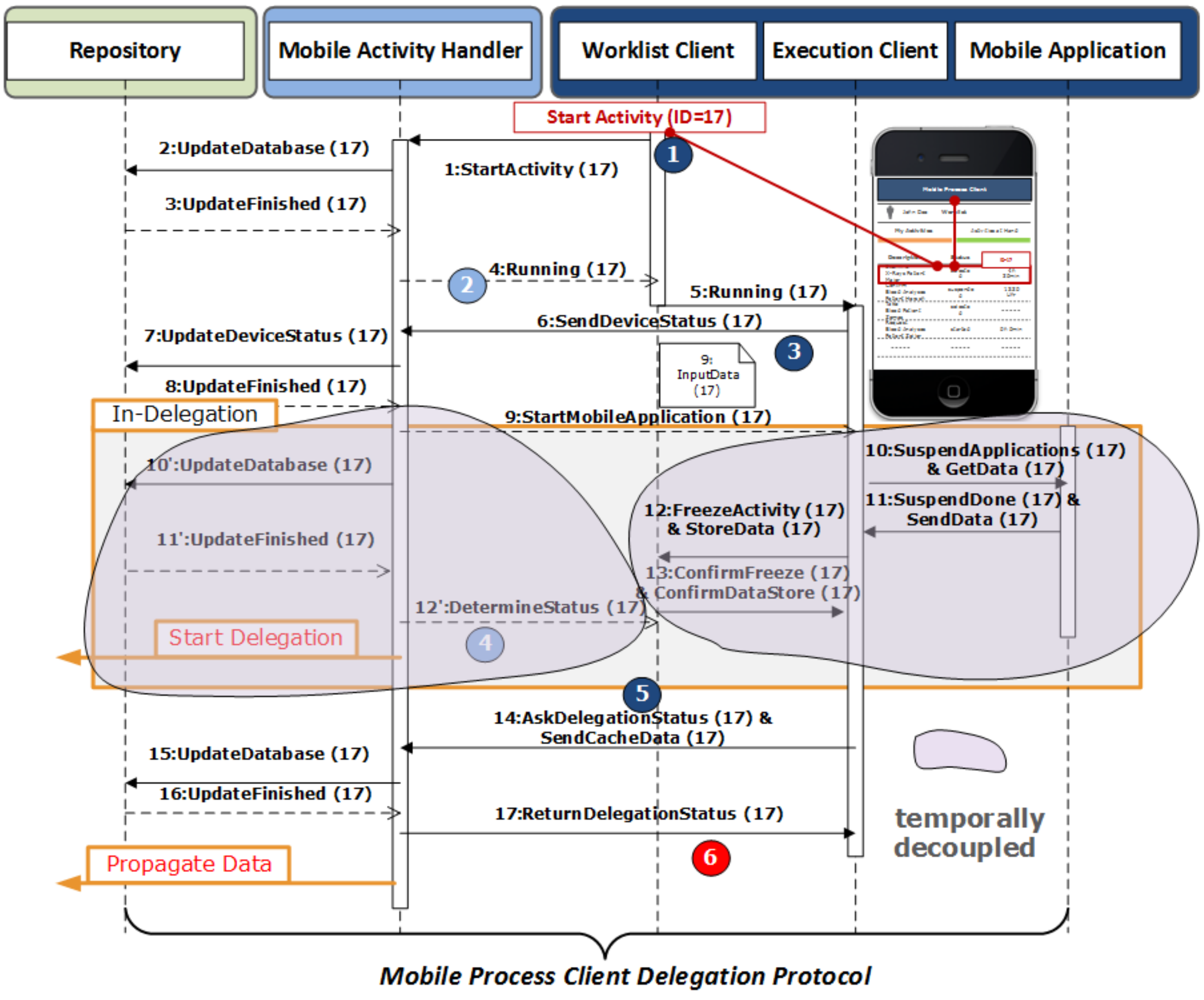

Figure 6. Delegation protocol

Adding Mobile Activities to Process Execution

For adding a mobile activity to a process model and hence integrating it with process execution, two fundamental solutions exist. These will be presented in this section. In particular, it will be shown how the challenges summarized in Table 4 are addressed by these solution approaches:

	Parameter	*Description*	*Type*	*Design Time*	*Instantiation Time*	*Activation Time*	*Delegation Time*
		Category I: Mobile device (SMD)					
Mobile Environment	SMD_{BS}	Battery Status	M	x	-	x	x
	SMD_{FF}	Form Factor	S	x	-	x	x
	SMD_{NT}	Network Type	M	-	-	x	x
	SMD_{GC}	Geometric Coordinate	M	-	-	x	x
		Category II: Mobile Activity (MA)					
Process	MA_{SC}	Symbolic Coordinate(s)	S	x	-	x	x
	MA_{GC}	Geometric Coordinate	M	-	-	x	x
	MA_{LR}	Location Range	S	x	-	x	x
	MA_{BS}	Battery Status	M	-	-	x	x
	MA_{U}	Urgency Value	S	x	-	x	x
	MA_{OFF}	Offline Mode	S	x	-	x	x
	MA_{FF}	Form Factor	S	x	-	x	x
	MA_{RF}	Response Frequency	S	x	-	x	x
	MA_{UT}	User Threshold	S	x	-	x	x
		Category III: Process (P)					
	P_{IST}	Instant Shutdown Threshold	S	x	-	x	x
		Category IV: Mobile User (MU)					
User Behavior	MU_{SC}	Symbolic Coordinate(s)	S	x	-	x	x
	MU_{IS}	Instant Shutdowns	S	-	-	x	x
		| Type: M=Measured, S=Symbolic | (x): relevant | (-): not relevant |					

Table 4. Parameters for processes with mobile activities

First, a backup service will be introduced, which adds a backup activity to ensure a robust execution of mobile activities. Second, a delegation service will be defined that automatically delegates the execution of mobile activities among available mobile users, if required. Before presenting these two services in detail, this section illustrates the basic steps required to add a mobile activity to process execution. Overall, the procedure encompasses four phases. Figure 8 shows in which of these phases manual steps (i.e., user interaction) and automated operations (i.e., delegation and backup service executions) are performed. Note that after creating an instance of a process model comprised of mobile activities, mobile users are not burdened with making decisions with regard to exceptions of a mobile activity. This behavior will be ensured since delegation and backup services as well as the user list management are performed automatically.

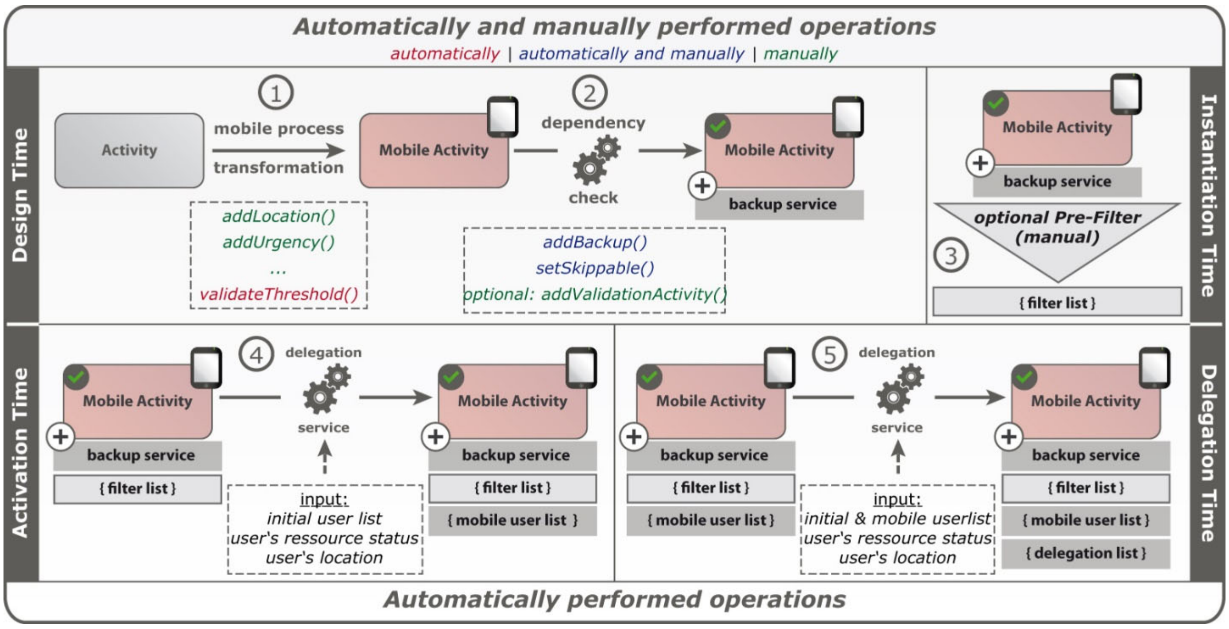

Figure 7. Procedure for integrating mobile activities into process execution

Design Time

The design of a process model which comprises mobile activities consists of two phases. During the first one, which is called *mobile process transformation phase (see Figure 7(1))*, a process designer flags selected activities of the given process model as *mobile*, i.e., these activities shall be executed by mobile users on their respective mobile devices during run time. In this context, the process designer determines parameters to each mobile activity *(see Figure 7(1))* that are presented in Table 4 relevant for the design time. In addition, a mobile activity threshold may be created. The latter defines the minimum number of users that shall be available at run time in order to execute this activity, i.e., the threshold allows controlling the assignment depending on the specific needs of the respective mobile processes. For all mobile activities, for which such a threshold is defined during this phase, the list of users who may perform the mobile activity is determined *(Figure 7(1), validateThreshold)* based on information stored in the user repository. Finally, activities whose chosen threshold value lies beyond the number of currently available users are highlighted to the process designer who may then alter this value.

The second design time phase is the *dependency check (see Figure 7(2))*. In this phase, it is determined for which mobile activities the backup service can be provided. While the *mobile process transformation* is done manually (except the validation of the threshold), the *dependency check* can be performed automatically. In this context, scenario specific dependency checks can be used. For example, a dependency check for mobile activities that write data elements, the following check will be automatically accomplished: First, all mobile activities are analyzed with respect to the data elements they provide for subsequent process activities. If a mobile activity writes such data, the backup service will be added for this mobile activity (see *Figure 7(2), addBackup)*. If this does not apply, the mobile activity may be skipped during run time without need for additional exception handling, i.e., operation *setSkippable* may be applied to such a mobile activity (see *Figure 7(2))*. Accordingly, attribute *IS_SKIPPABLE* of this mobile activity is set to *true*. While the backup service (or *setSkippable* operation) is automatically added to a mobile activity, the third operation of this phase (*Figure (2)*, *addValidationActivity)* is performed manually, i.e., the process designer must decide whether or not the execution of backup activities must be manually confirmed during run time. In order to enforce this behavior, the process designer activates the validation activities set by the backup service, i.e., the sync flag read by the validation activities will be set to *true*.

To understand the next three phases with respect to design and run time, Figure 8 summarizes the lifecycle of a mobile activity. Thereby, it relates all phases shown in Figure 7 with the lifecycle (see *1-4 in* Figure 8).

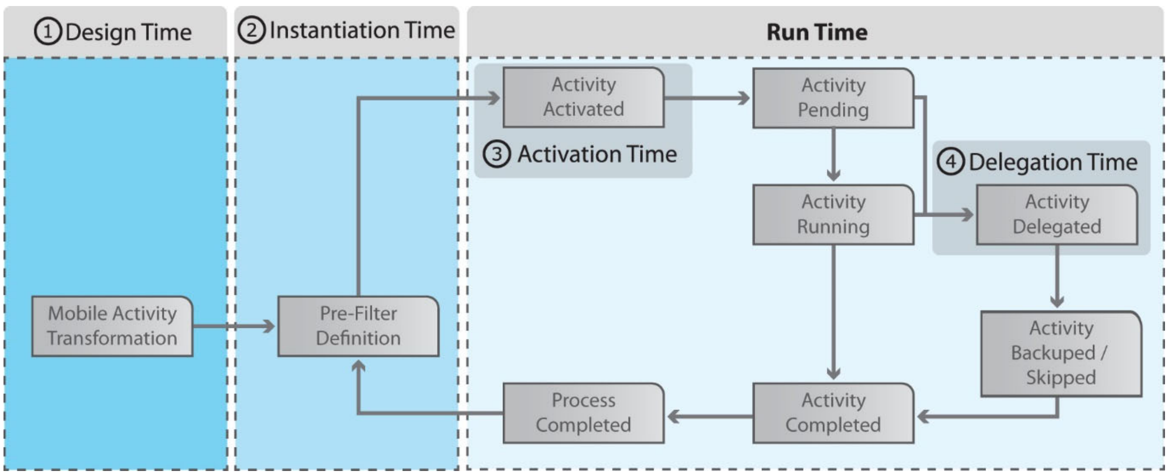

Figure 8. Mobile activity lifecycle

Instantiation Time
When creating a process instance, a service is provided to change the run time configuration of this instance *(see Figure 7(3), addFilterList)*. This service aims to cope with the dynamics of mobile environments. To perform such a change, the following steps are applied. First, for all mobile activities, user lists are computed. Thereby, only currently online users are considered. Second, for each mobile activity it must be decided whether to change its location or urgency. In addition, users authorized to execute other *activities* may be removed. The latter option allows covering different kinds of mobile business scenarios properly. For example, the mobile device of a physician who needs to cope with an emergency should not be the target for upcoming mobile activities.

Delegation Time
When delegating a mobile activity at run time, it is automatically delegated to another user possessing same rights. Further, for each mobile activity, a delegation list is managed. This list will be created after the first delegation becomes necessary. It also stores a history of all delegations for this mobile activity.

Generally, the following issues are crucial with respect to mobile activity execution (see Figure 11):

- An execution exception of a mobile activity that produces data that is consumed by subsequent process steps may cause severe run time exceptions (Reichert et al., 1999; Reichert & Weber, 2012) (see *missing data in Figure 9*).

- An execution exception of a mobile activity might cause a deadlock (see *deadlock in Figure 9*), i.e., if a mobile activity cannot be properly completed, succeeding activities might not be activated.
- Regarding mobile activity execution, usually, a time period is specified indicating the maximum duration of this activity. For example, it might be required that a blood test is finished within 5 minutes. Accordingly, any execution exception of a mobile activity should be handled in time in order to meet respective temporal constraints.

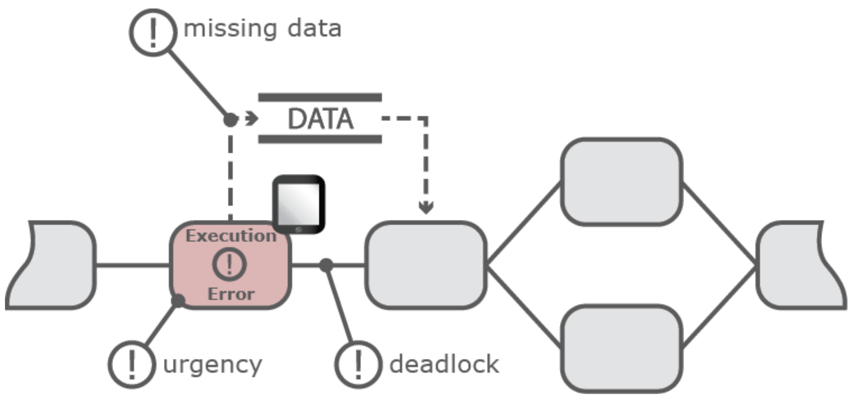

Figure 9. Mobile activity execution challenges

SERVICE-ORIENTED SOLUTION APPROACH

The solution approach towards the robust execution of mobile activities, which is presented in this section, tackles the challenges introduced before. Table 5 gives an overview of the main challenges by showing which service, i.e., mobile process transformation (*MPT*), backup service (*BS*), list management (*LM*), and mobile delegation service (*MDS*) addresses which of these challenges.

Challenge	Component	Description
Connectivity	LM MDS	Only connected devices will be added to the user and delegation lists. The MDS refreshes these lists continuously and accomplishes any delegation required when a mobile device loses its connection.
Resource Behavior	LM MDS	Only mobile users having a sufficient resource behavior will be added to the user and delegation lists.
User Location	LM	If a location X is explicitly assigned to m*obile activity*, only users whose current location matches X are added to the respective user lists.
Data Dependencies	BS	The backup service ensures that exceptions during the execution of a m*obile activity* do not harm overall process execution, e.g., if subsequent activities are data-dependent on the failed mobile activity, the latter will be replaced by a respective backup activity added to the process model.
Location	MPT	A m*obile activity* may require a certain location for its execution.

Urgency	MPT MDS BS	The urgency of an activity may be set at design time. In turn, the MDS then utilizes this information as trigger for delegating the mobile activity, i.e., a backup service ensures that the mobile activity will be always executed within the specified time frame.

Table 5. Parameters and solution components of mobile activities

Delegation Service

During mobile activity execution, the *mobile delegation service* (MDS) ensures that already assigned mobile activities are automatically re-delegated to another authorized mobile user in case of exceptions. Since this delegation service maintains several user lists, the latter are first summarized before presenting the MDS.

User List Management

To enable a flexible delegation and hence to foster robust execution of a mobile activity *t*, three user lists are maintained for it: ul_{init}, ul_{mob}, and dl_{mob}. User list ul_{init} contains all mobile users that are, in principle, authorized to perform mobile activity *t*. Based on ul_{init},the mobile user list ul_{mob} is determined. Thereby, a mobile user from ul_{init} is only added to ul_{mob}, if the user is currently online, user's location complies with the one of *t*, and the user is not excluded by any filter defined at instantiation time. Based on ul_{mob}, *t is assigned to available* mobile users. Furthermore, if *t* shall be delegated, a mobile delegation list dl_{mob} is determined. First of all, all users contained in ul_{mob} are added to dl_{mob}. Then, dl_{mob} is ordered by taking the resource behavior of mobile users into account. A low priority is assigned to a mobile user if his resource behavior is inappropriate. Both lists ul_{mob} and dl_{mob} are re-calculated when the connectivity status of a user from list ul_{init} changes.
Overall, when considering these three lists, the mobile delegation service may enter six different states (see Figure 10). The latter are denoted as *t(<STATE>)* and respective state transitions as T_i. Note that the delegation service starts when a mobile activity *t* becomes activated. The delegation service will be activated before executing the respective activity only if the mobile activity has a value set for *urgency* and no authorized mobile user will promptly execute the activity, then it will be delegated to an authorized mobile user, i.e., delegation is used for changing activity state to running.

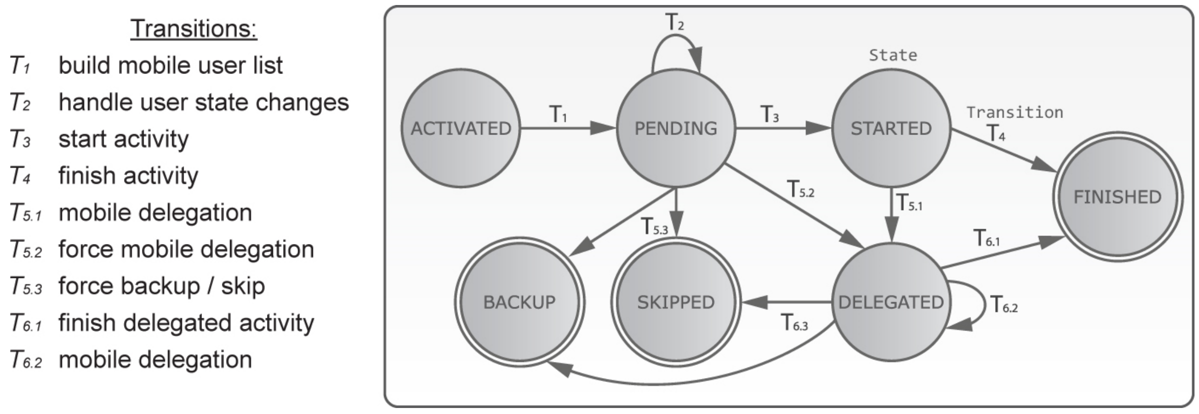

Figure 10. Mobile delegation service flow during run time

The scenarios shown in Table 6 are relevant when taking urgency (specified period of time when the activity must be finished) to_u ($to_u = 0$ denotes a timeout), user list threshold th_{mul}, and the ability to skip a mobile activity *t* into account.

Scenario	Description	State Chain
Normal activity execution	$user_a \in ul_{mob}$ starts mobile activity t and performs it.	$t(PENDING) \rightarrow T_3 \rightarrow t(STARTED) \rightarrow T_4 \rightarrow t(FINISHED)$
Delegated activity execution	$user_a \in ul_{mob}$ starts mobile activity t. Then user state changes to offline or $to_u = 0$ holds, such that t will be automatically delegated to another user $user_b \in ul_{mob}$ who finishes the mobile activity.	$T_3 \rightarrow t(STARTED) \rightarrow T_{5.1} \rightarrow t(DELEGATED) \rightarrow T_{6.1} \rightarrow t(FINISHED)$
Forced delegation	A forced delegation becomes necessary if the following holds: $t(PENDING) \wedge (\|ul_{mob}\| < th_{mul} \vee to_u = 0) \vee t(DELEGATED) \wedge (to_u = 0 \vee State(user_b)$ changes to offline) ➔ t must be delegated to another user $user_n \in ul_{mob}$.	$T_{5.2} \rightarrow t(DELEGATED) \vee T_{6.2} \rightarrow t(DELEGATED)$
Skip or Backup	Skip or backup will be performed if the following holds: If $t(PENDING) \wedge to_u = 0 \wedge \|ul_{mob}\| = 0) \vee (t(DELEGATED) \wedge to_u = 0 \wedge \|dl_{mob}\| = 0)$. Furthermore, if IS_SKIPPABLE(T)=true, t will transit to SKIP, otherwise to BACKUP	$t(PENDING) \rightarrow T_{5.3} \rightarrow t(SKIP) \vee t(BACKUP)/ t(DELEGATED) \rightarrow T_{6.3} \rightarrow t(SKIP) \vee t(BACKUP)$

Table 6. Scenarios in which mobile delegation service is applied

Backup Service

A particular challenge arises if no mobile user is available for processing an activated mobile activity that produces data during run time, i.e., if no delegation is possible anymore. In order to ensure that these mobile activities can still be processed, a backup service is provided. Basically, it consists of two operations, which are added to a process fragment replacing the mobile activity in case of exceptional situations. The first one is called *simple backup operation*, while the second is called *complex backup operation*. Before presenting the backup service in detail, three questions have to be briefly discussed with respect to the backup service: (1) Since the backup operation is performed on a stationary computer and hence failures are more unlikely than using a mobile device, the question arise why performing the activity on a mobile device at first? One can argue that in this case the mobile device is not necessary and affects overall robustness. (2) How are skipping of mobile activities and the backup service relate to each other? (3) Does it make sense to provide the validation activity with the backup service since it may block execution and resulting in the same exceptional situation as caused before by the mobile device? The first two questions will be discussed in the following, while the third one will be discussed after presenting the simple backup operation.

First of all, see Figure 11. Note that the backup service is only applied to mobile activities which write data or have other specifically defined attributes. In turn, the case using the backup service in the context of mobile activities that write data is the one mainly addressed. As depicted in Figure 11, in the best and average case, a normal execution or delegation prevents using the backup service. Note that in case no more delegation is possible, i.e., no more suitable mobile user is able to perform this activity. Only then, the backup operation becomes necessary. Question 2 can now be answered based on this. Skipping an activity and the backup service are not related. Activities to which the backup service is applied can only be delegated or finalized based on the backup service. In general, skipping an activity is only possible for those activities that do not write data. Regarding Question 1, a mobile activity is intended to be performed in a mobile manner. Since the backup operation constitutes a very final alternative, originally intended behavior is preserved more properly. For example, assume the following situation: Two physicians work on urgent mobile activities with their mobile devices. Then, both devices run out of battery (frequently observed in the context of the above presented MEDo project). Further, no other physicians are able to perform these mobile activities (frequently observed as well). In this case, using stationary computers to complete the activities are highly welcome. For this purpose, the backup service will be used. Due to the lack of space, the way how mobile activities are adjusted to be executed on stationary computer can be obtained from (Pryss & Reichert, 2016).

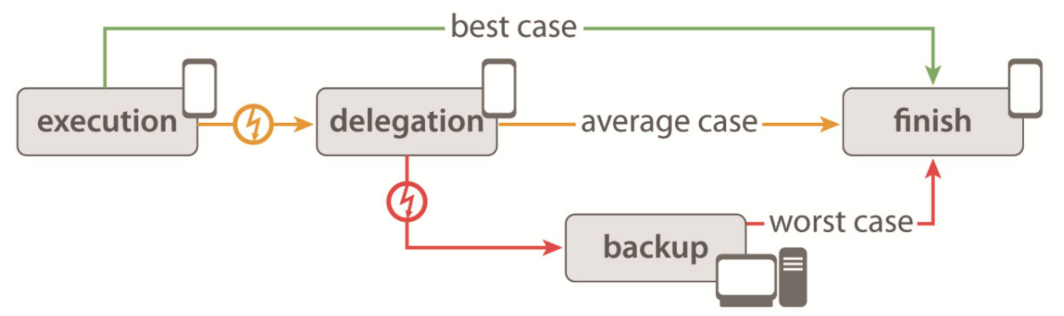

Figure 11. Use case of backup operation

CURRENT RESEARCH

Among others, our current research on mobile activity support deals with the following issues. In general, certain constraints may have to be obeyed when executing mobile activities. As example, consider entailment constraints that may exist between different mobile activities. When executing a mobile process, for example, it might be desirable that two mobile activities are executed by the same user. Related research on integrating such constraints with business processes has received growing attention recently. However, realizing entailment constraints in the context of mobile processes and activities raises additional issues, which must be integrated with our backup and delegation services. Furthermore, a way to specify rules for enhancing the delegation service must be developed; e.g., users should be allowed to specify their own delegation rules.

CONCLUSION

This paper presented an approach for enabling business processes with mobile activity support. The backup service as well as the mobile delegation service allow for a robust process execution. For this purpose, four fundamental issues need to be considered. First, the specific challenges of executing process activities in a mobile environment must be well understood. Second, these challenges must be properly addressed at both design and run time. Third, mobile activities must be executed in a robust way, the backup service and mobile delegation services foster such robustness with respect to mobile activity execution. Fourth, user acceptance is crucial in the context of mobile activity support. Accordingly, the presented services do not involve mobile users in exception handling directly. Finally, a sophisticated architecture has been described showing how the presented approach can be realized in a service-oriented environment.

REFERENCES

van der Aalst, W.M.P., & Weske, M. (2001). The P2P approach to interorganizational workflows. In Advanced Information Systems Engineering (pp. 140-156). Springer Berlin Heidelberg.

Alonso, G., Agrawal, D., El Abbadi, A., Kamath, M., Gunthor, R., & Mohan, C. (1996). Advanced transaction models in workflow contexts. In Data Engineering, 1996. Proceedings of the Twelfth International Conference on (pp. 574-581). IEEE.

Alonso, G., & Schek, H. J. (1996). Research issues in large workflow management systems. In Proceedings of NSF Workshop on Workflow and Process Automation in Information Science.

Alonso, G., Gunthor, R., Agrawal, D., El Abbadi, A., & Kamath (1995). Exotica/FMDC: Handling disconnected clients in a workflow management system.

Ayora, C., Torres, V., Reichert, M., Weber, B., & Pelechano, V. (2013). Towards run-time flexibility for process families: open issues and research challenges. In Proc. Business Process Management Workshops (pp. 477-488).

Barros, A., Dumas, M., & Oaks, P. (2005). A critical overview of the web services choreography description language. BPTrends Newsletter, 3, 1-24.

Battista, D., & Leoni, M., & Gaetanis, A., & Mecella, M., & Pezzullo, A., & Russo, A., & Saponaro, C. (2008). ROME4EU: A Web service-based process-aware system for smart devices. In: Proc. ICSOC'08 (pp. 726-727).

Bauer, T., Reichert, M., & Dadam, P. (2003). Intra-subnet load balancing in distributed workflow management systems. Int'l Journal of Cooperative Information Systems, 12, (pp. 205-223).

Becker, C., & Dürr, F. (2005). On location models for ubiquitous computing. Personal and Ubiquitous Computing, 9(1), 20-31.

Chen, N., Cardozo, N., & Clarke, S. (2016). Goal-driven service composition in mobile and pervasive computing. IEEE Transactions on Services Computing.

Cichocki, A., & Rusinkiewicz, M. (1998). Migrating workflows. In Workflow Management Systems and Interoperability (pp. 339-355). Springer Berlin Heidelberg.

Crampton, J., & Khambhammettu, H. (2008). Delegation and satisfiability in workflow systems. In Proc. of the 13th ACM symposium on Access control models and technologies (pp. 31-40).

Gaaloul, K., & Charoy, F. (2009). Task delegation based access control models for workflow systems. In Software Services for e-Business and e-Society (pp. 400-414).

Geiger, P., Pryss, R., Schickler, M., & Reichert, M. (2013). Engineering an Advanced Location-Based Augmented Reality Engine for Mobile devices. Technical Report, University of Ulm.

Ghalimi, I. C., & CEO, I. (2006). BPM 2.0.

Hackmann, G., Haitjema, M., & Gill, C. (2006). Sliver: A BPEL workflow process execution engine for mobile devices. In: Proc. ICSOC'06 (pp. 503-508).

Hahn, K. & Schweppe, H. (2009). Exploring transactional service properties for mobile service composition. In: Proc. MMS'09 (pp. 39-52).

IBM Mobile Business Process Management (IBM), http://www.redbooks.ibm.com/abstracts/sg248240.html?Open, 2014, [Online; accessed 12-May-2018].

Jones, V.M., van Halteren, A.T., Dokovski, N.T., Koprinkov, G.T., Peuscher, J., Bults, R.G.A., Konstantas, D., Widya, I.A., & Herzog, R. (2006). Mobihealth: mobile services for health professionals. Technical Report TR-CTIT-06-38, Enschede.

Kunze, C.P. (2005). Demac: A distributed environment for mobility-aware computing. In Adjunct Proc. of the Third International Conference on Pervasive Computing (pp. 115-121).

Lenz, R., & Reichert, M. (2007). IT Support for Healthcare Processes -Premises, Challenges, Perspectives. Data Knowl. Eng., 61(1), pp. 39–58.

Marinescu, A., Dusparic, I., Taylor, A., Cahill, V., & Clarke, S. (2015). P-marl: Prediction-based multi-agent reinforcement learning for non-stationary environments. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (pp. 1897-1898). International Foundation for Autonomous Agents and Multiagent Systems.

Martin, D., Wutke, D., & Leymann, F. (2008). A novel approach to decentralized workflow enactment. In Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE (pp. 127-136). IEEE.

Meister, V.G. (2011). Geschäftsregelbasierte Ansteuerung betrieblicher Anwendungssysteme am Beispiel der Open Source Process Engine Activiti. Betriebliche Anwendungssysteme, 65.

Philips, E., van Der Straeten, R., & Jonckers, V. (2011). NOW: Orchestrating services in a nomadic network using a dedicated workflow language. Science of Computer Programming.

Pryss R., Tiedeken J., & Reichert M. (2010). Managing Processes on Mobile Devices: The MARPLE Approach. In Proc. CAiSE'10 Demos.

Pryss, R., Musiol, S., & Reichert, M. (2013). Collaboration Support Through Mobile Processes and Entailment Constraints. In Proc. 9th IEEE Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing.

Pryss, R., Tiedeken, J., Kreher, U., & Reichert, M. (2010). Towards Flexible Process Support on Mobile Devices. In Proc. CAiSE'10 Forum (pp. 150–165).

Pryss, R., Mundbrod, N., Langer, D., & Reichert, M. (2015). Supporting medical ward rounds through mobile task and process management. Information Systems and e-Business Management, 13(1), 107-146.

Pryss, R., Reichert, M., Schickler, M., & Bauer, T. (2016). Context-Based Assignment and Execution of Human-centric Mobile Services. In Mobile Services (MS), IEEE International Conference on (pp. 119-126). IEEE.

Pryss, R., Geiger, P., Schickler, M., Schobel, J., & Reichert, M. (2016). Advanced Algorithms for Location-Based Smart Mobile Augmented Reality Applications. Procedia Computer Science, 94, 97-104.

Pryss, R., & Reichert, M. (2016). Robust execution of mobile activities in process-aware information systems. International Journal of Information System Modeling and Design, 7(4), 50-82.

Reichert, M., Dadam, P., Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., ... & Goeser, K. (2009). Enabling Poka-Yoke workflows with the AristaFlow BPM Suite.

Reichert, M., & Weber, B. (2012). Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer.

Reichert, M., & Weber, B. (2013). Process Change Patterns: Recent Research, Use Cases, Research Directions. In Seminal Contributions to Information Systems Engineering - 25 Years of CAiSE, (pp. 398–404).

Reichert, M., Bauer, T., & Dadam, P. (2009). Flexibility for distributed workflows.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. Computer, *29*(2), 38-47.

Schaad, A. (2003). A Framework for Organisational Control Principles. PhD thesis, The University of York, York, England.

Schmidt, H., & Hauck, F.J. (2007). SAMPROC: middleware for self-adaptive mobile processes in heterogeneous ubiquitous environments. In: Proc. 4th Middleware Doctoral Symposium (pp. 1-6).

Schmidt, H., Kapitza, R., & Hauck, F.J. (2007). Mobile-process-based ubiquitous computing platform: a blueprint. In: Proc. 1st Workshop on Middleware-application interaction (pp. 25-30).

Schobel, J., Schickler, M., Pryss, R., Nienhaus, H., & Reichert, M. (2013). Using Vital Sensors in Mobile Healthcare Business Applications: challenges, Examples, Lessons Learned. In Proc. 9 Int'l Conference on Web Information Systems and Technologies (pp. 509–518).

Schobel, J., Pryss, R., Wipp, W., Schickler, M., & Reichert, M. (2016). A Mobile Service Engine Enabling Complex Data Collection Applications. In International Conference on Service-Oriented Computing (pp. 626-633). Springer International Publishing.

Schobel, J., Pryss, R., Schickler, M., & Reichert, M. (2016). A Lightweight Process Engine for Enabling Advanced Mobile Applications. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 552-569). Springer International Publishing.

Schobel, J., Pryss, R., Schickler, M., Ruf-Leuschner, M., Elbert, T., & Reichert, M. (2016). End-user programming of mobile services: empowering domain experts to implement mobile data collection applications. In: IEEE 5th International Conference on Mobile Services (pp. 1-8). IEEE.

Stürmer, G., Mangler, J., & Schikuta, E. (2009). Building a modular service oriented workflow engine. In Proc. Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on (pp. 1-4).

Tuysuz, G., & Avenoglu, B., & Eren, P. E. (2013). A Workflow-Based Mobile Guidance Framework for Managing Personal Activities. In Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 Seventh International Conference on (pp. 13-18). IEEE.

Wakholi, P. K., & Chen, W. (2012). Workflow Partitioning for Offline Distributed Execution on Mobile Devices. Process Aware Mobile Systems. Applied to mobile-phone based data collection.

Wodtke, D., & Weikum, G. (1997). A formal foundation for distributed workflow execution based on state charts. In Database Theory—ICDT'97 (pp. 230-246). Springer Berlin Heidelberg.

Workflow Management Coalition (WFMC), http://www.wfmc.org/, 2014, [Online; accessed 12-May-2018].

Zaplata, S., Dreiling, V., & Lamersdorf, W. (2009). Realizing mobile Web services for dynamic applications. In Proc. I3E'09 (pp. 240-254).

Zaplata, S., Hamann, K., Kottke, K., & Lamersdorf, W. (2010). Flexible execution of distributed business processes based on process instance migration. Journal of Systems Integration, 1(3), (pp. 3-16).

Zaplata, S., Kottke, K., Meiners, M., & Lamersdorf, W. (2010). Towards run time migration of WS-BPEL processes. In: Proc. WESOA'09.