Towards Flexible Process Support on Mobile Devices

Rüdiger Pryss, Julian Tiedeken, Ulrich Kreher, and Manfred Reichert

Institute for Databases and Information Systems, Ulm University, Germany
{ruediger.pryss,julian.tiedeken,ulrich.kreher,manfred.reichert}@uni-ulm.de

Abstract. Ubiquitous computing is considered as enabler for linking everyday life with information and communication technology. However, developing pervasive and mobile applications that provide personalized user assistance still constitutes a challenge. Mobile application scenarios are diverse and encompass domains like healthcare, logistics, and sales. For their support two fundamental technologies with increasing maturity are emerging: *development frameworks for mobile devices* and *light-weight process engines*. Their integrated use, however, is in a rather premature state. Generally, the use of a process engine for supporting mobile collaboration raises many challenging issues. This paper picks up some of these challenges and shows how we have coped with them in the MARPLE project. MARPLE targets at a tight integration of process management technology with mobile computing frameworks in order to enable mobile process support in advanced application scenarios. We give insights into the MARPLE architecture and its components. In particular, we introduce the MARPLE process engine, which enables light-weight as well as flexible process support on mobile devices. This will be key for mobile user assistance in advanced application scenarios.

Keywords: Mobile Process, Distributed Process, Process Management.

1 Introduction

Mobile assistance in daily life as empowered by information and communication technology is a much discussed topic [1,2]. To better understand relating challenges, we analyzed real-world process scenarios from different domains in which mobile user assistance is urgently needed. This includes process scenarios from healthcare, logistics, and sales. Altogether, our studies revealed the fundamental role of process support in the context of mobile and personalized user assistance. In this paper we pick up a realistic healthcare scenario in which chronically ill patients shall be assisted by mobile devices. The latter shall give recommendations about therapies (e.g., medications), collect data from the patient, or help general practioners to plan encounters with their patients. Recommendations may be made remotely by healthcare professionals and often depend on previously gathered patient data (e.g., blood pressure or blood sugar). Despite its high potential, so far, there exists no comprehensive mobile user assistance for realizing such scenarios in a flexible way and in a large scale. One major task in

P. Soffer and E. Proper (Eds.): CAiSE Forum 2010, LNBIP 72, pp. 150–165, 2010.

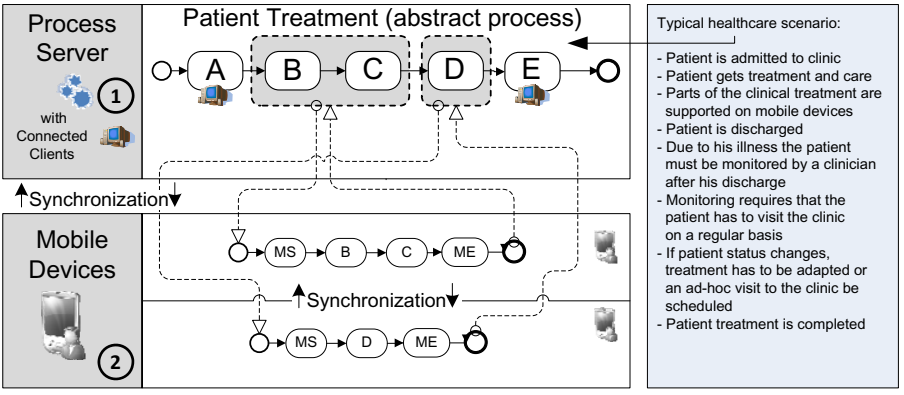

Fig. 1. Abstract Healthcare Scenario (see Fig. 2 for a connect scenario)

this context is to decide which parts of a global process (i.e. *process fragments*) shall run on mobile devices (e.g., a patient's mobile) and which ones shall be controlled by process servers (e.g., running on the physician's workstation or any other backend server). In the following we refer to such a healthcare scenario to discuss fundamental challenges, and to show the benefits coming with mobile user assistance. Fig. 1 illustrates both a traditional realization of this scenario (①) and its implementation based on mobile user assistance (①+②). After discharging patients the usual way to monitor their health status is to schedule regular visits for them in the clinic. In certain cases, however, this can lead to delayed adaptations of treatment plans, e.g., if patient status changes while they stay at home. To improve this situation and to enable remote monitoring of patients (cf. Fig. 1 Steps B,C), mobile data collection and mobile user assistance ② would be highly welcome by all parties; i.e., patients should be assisted by a mobile device which gathers medical data from them and informs clinicians about important status changes. Further, it should enable clinicians to remotely adapt their treatment plan or to schedule meetings with patients if required. In addition, it might be desirable to run parts of the treatment procedure on mobile devices (cf. Fig. 1 Step D). To realize such scenario and to provide personalized treatment support, patient-specific application logic needs to be provided on the mobile device. As a consequence, the overall treatment process is partitioned ② and the resulting process fragments are deployed to both process servers and mobile devices. In particular, process fragments running on mobile devices need to be quickly configurable to the specific patient and be dynamically adapted if patient's status changes. Hard-coded process implementations are therefore not adequate in this context. Instead flexible support of processes with mobile assistance is needed.

To enable mobile process assistance we developed a light-weight process engine called *MARPLE* [1] that runs on mobile devices and is able to interact with backend processes if required. In addition, we provide advanced tools for defining,

[1] **MA**naging **R**obust mobile **P**rocesses in a comp**LE**x world.

configuring, verifying and deploying process fragments in such an environment. In the following, we focus on fundamental challenges on the one hand and on the core architecture and components of the *MARPLE* mobile process engine on the other hand. Conceptual issues related to the partitioning of processes as well as to the synchronization of the resulting process fragments are outside the scope of this paper. When developing the *MARPLE* engine we had one shining example in mind - the ADEPT process management system, which we had developed during the last decade [3]. In particular, we adopted basic design principles from ADEPT (e.g., its correctness-by-construction principle and its dynamic process change capabilities), but also aligned the *MARPLE* architecture with specific needs relating to mobile process support.

The remainder of this paper is organized as follows: Section 2 introduces a concrete application scenario. In Section 3 we describe requirements derived from case studies as basis for the *MARPLE architecture*. Section 4 gives insights into the MARPLE architecture, while Section 5 shows how the described application scenario can be supported. In Section 6 we discuss selected *MARPLE* features in the context of advanced application scenarios. Section 7 discusses related work and Section 8 concludes with a summary and outlook.

2 Application Scenario

Fig. 2 shows a typical healthcare process (modeled in terms of BPMN) involving three parties: clinic, patient, and homecare. The first swim lane shows activities as performed in the clinic, which also starts the process. When completing Step ①, the execution of the process fragment on the mobile device of the patient is triggered (patient swimlane). This mobile process then collects medical data from the patient and coordinates required actions; e.g., to measure blood pressure or to gather ECG data.

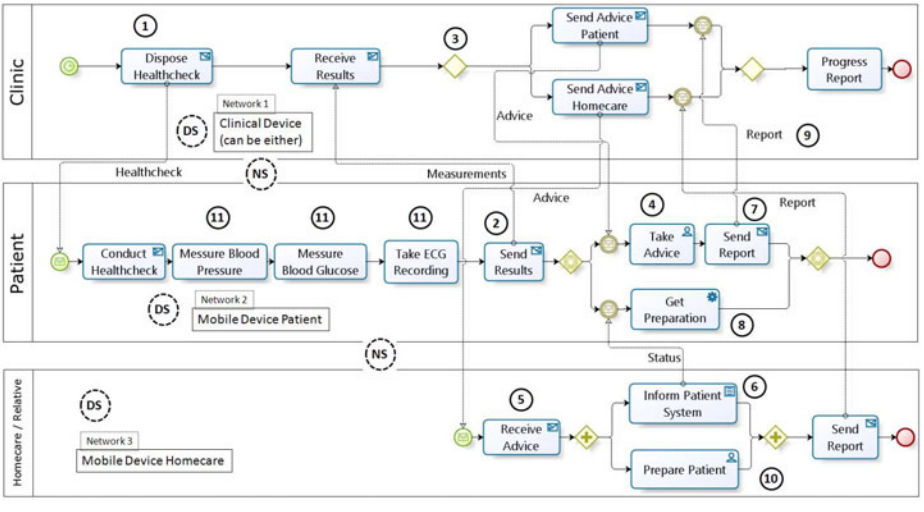

Fig. 2. Healthcare Process with Mobile Patient Assistance

Let us consider the patient-specific process fragment in more detail: While the patient stays at home, he gets a message from the clinic through his mobile device. Then he measures and collects the requested data being assisted by the process fragment running on his mobile device. Following this, results are sent back to the clinic ②, which then decides about next steps ③. Ideally, everything is OK and no special actions concerning the patient are required. In this case a message is sent back to the mobile device containing information about the patient's medication ④. In case of problems, the clinic might send an alternative message with information about further or special treatment to homecare ⑤ (either provided by a professional service or by relatives of the patient). In the latter case, an additional process fragment is started on the mobile device of the person being responsible for homecare. Its execution then has to be synchronized with the one running on the patient's mobile device. Finally, either the process running on the mobile device of homecare or the one of the patient sends back a report ⑦ to the clinic, before the process is finished.

3 Requirements

Table 1 gives an overview of characteristic requirements raised when running processes and process fragments on mobile devices. We elicitated these requirements when analyzing different process scenarios from domains like healthcare, sales, logistics, and emergency management.

Table 1. Requirements Overview

Requirements for Enabling Mobile Processes	
Category I: Process Implementation	**Category III: Runtime**
R_1 ***(Partitioning):*** It must be possible to partition a global process model and to allocate the resulting fragments on mobile devices and (distributed) process servers.	R_7 ***(Synchronization):*** When running fragments on distributed machines and mobile devices respectively, their execution must be correctly synchronized. Further, messages must be exchanged in a reliable and secure way. A mechanism for queuing messages is needed.
R_2 ***(Soundness):*** Soundness (i.e., correct execution behavior) of both the global process (i.e., the process choreography) and the process fragments (i.e., process orchestrations) needs to be ensured.	R_8 ***(Adaptations):*** Both the global process model and its constituting fragments might have to be adapted during runtime (e.g., to deal with exceptions). Such adaptations must not lead to inconsistencies or errors.
R_3 ***(Lifecycle):*** Full lifecycle support for the global process as well as its constituting process fragments is needed.	R_9 ***(Hand-Over):*** Handing over running process fragments between mobile devices must be possible; i.e., to move a running process fragment from the current device to a new one and to restart its execution from a safe point.
Category II: Supporting Infrastructure	R_{10} ***(Robustness/Self-Healing):*** Mobile processes need to be executed in a robust manner. Self-healing mechanisms for reacting on unforeseen events (e.g., device failures) are required.
R_4 ***(Configuration Management):*** The infrastructure must provide user-friendly mechanisms for configuring process and service deployment, device characteristics, and application components.	
R_5 ***(Connection Management):*** Physical problems like broken connections or mal-functioning devices need to be handled by the supporting infrastructure, but without burdening users.	R_{11} ***(Real-Time Data):*** A mobile process must be able to combine gathered real-time data from sensors (e.g., blood pressure of a particular patient). Furthermore, failure data (e.g., a sensor does not deliver data) and QoS parameters (e.g., execution time or cost) need to be considered.
R_6 ***(User Management):*** The infrastructure must enable user changes on the mobile device without necessitating users to be aware of the location and characteristics of the mobile device (or surrounding sensors). Additionally, the infrastructure must provide context-aware mechanisms to decide which user change is possible and suitable. If a user must skip his activity (e.g., due to an emergency call), the infrastructure must cope with these unforeseen situations as well. Therefore a categorization of user changes must be provided by the infrastructure.	R_{12} ***(Events):*** During process execution events have to be gathered, filtered, aggregated, and processed. In addition, a categorization of events is needed for enabling better configuration support for devices and processes.
	R_{13} ***(Actor Assignments):*** Actor assignments for process steps and process fragments respectively must be resolvable at any point in time during process execution. Furthermore, actor assignments of upcoming process steps should be resolved in advance in order to pursue an optimal migration strategy.

Reconsider our example from Section 2. In this simple scenario three process fragments exist whose execution needs to be synchronized among the three parties. In this context, the overall system architecture must be able to cope with communication problems, device failures, and so forth. In Fig. 2 the pictograms with label DS and NS indicate potential network and device switches within the overall process choreography. For example, assume that the mobile device of the patient or its connection with the clinic fail. Consequently, the clinic has no information about the status of the patient, but only knows that the network connection is broken. Such failure scenarios must be adequately covered by the architecture.

4 MARPLE - Overview and Architecture

We first describe the *MARPLE architecture* (cf. Fig. 3) whose two core components are the *MARPLE Mobile Engine* and the *MARPLE Mediation Center*. We focus on those parts of the *MARPLE architecture* that are fully implemented and relevant in the context of our application scenario. Other components of MARPLE are only sketched and will be subject of future papers.

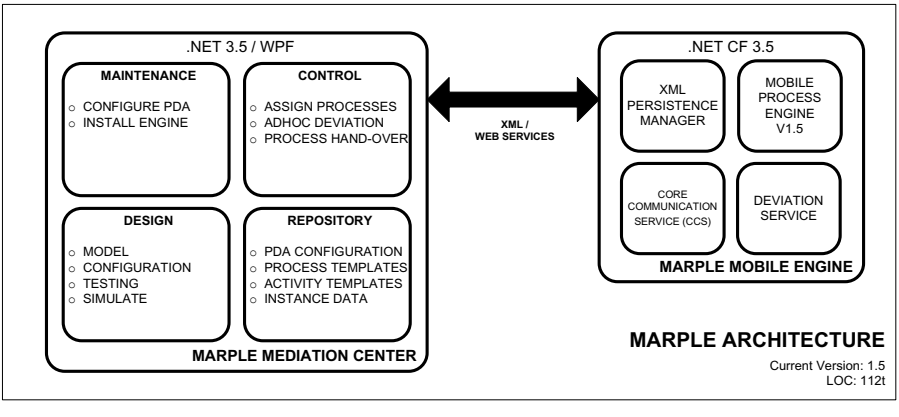

Fig. 3. MARPLE Architecture

4.1 MARPLE Mediation Center

The *MARPLE Mediation Center* (cf. Fig. 3) consists of four major parts. First, its *Maintenance* component enables configuration of mobile devices such that they can be used for mobile process support. Second, the *Control Unit* enables users to assign executable process fragments to mobile devices, to enact them on the mobile device, to invoke user forms, web services and other kinds of activity components during process fragment execution, and to define ad-hoc deviations from the prescribed process logic if required. Another fundamental feature of the MARPLE *Control Unit* is its ability to hand-over running process fragment instances from one mobile device to another; e.g., if a patient wants to switch his

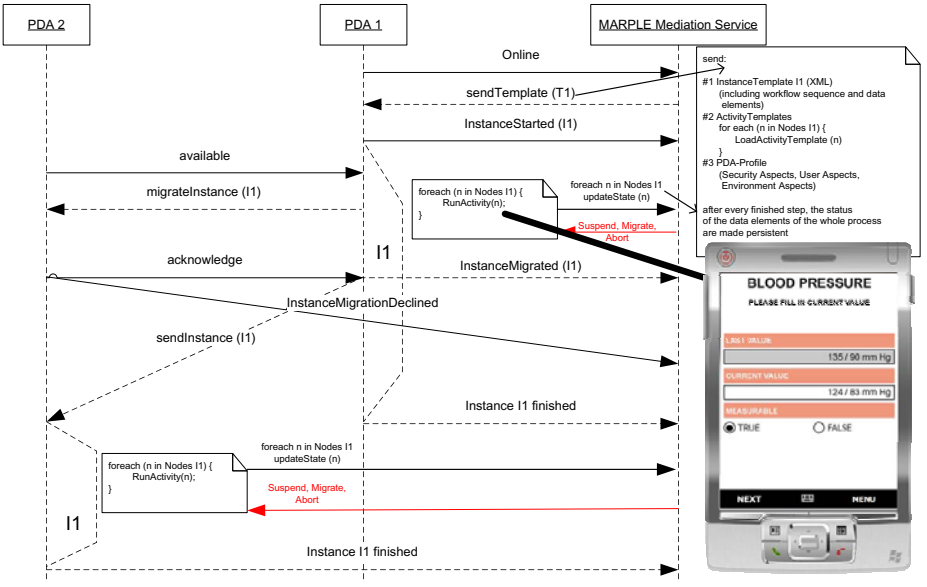

Fig. 4. MARPLE: Interaction Sequence

device or a homecare person wants to hand over his process fragment to someone else. Like in the case of ad-hoc changes, such hand-over can be initiated locally by the owner of the mobile device as well as remotely by authorized users via the *Control Unit*.

Fig. 4 exemplarily illustrates possible interactions between the *MARPLE Mediation Service* and two mobile devices. Initially, only one mobile device is involved in the interaction. Then a second device is added. Following this, the process instance running on the first mobile device is handed over to the newly introduced one, e.g., due to connection problems with the first device, better technical features of the new one, or needed hand-over of tasks. As mentioned this handing over can be triggered either by the *MARPLE Mediation Center* or by the device owners. During process execution, the *Control Unit* may suspend, resume, abort, and monitor running processes. Finally, *MARPLE Mobile Process Engine V1.5* logs process events using the *Persistence Manager* (cf. Fig. 3). Another important aspect that emerges when integrating end-users and process management technology on mobile devices concerns usability. We provide advanced support for deploying the *MARPLE* process engine to mobile devices. The *Mediation Center* makes the engine available through web services, which enables users to deploy it to their mobile devices.

4.2 MARPLE Modeler

Another important component of the *MARPLE Mediation Center* is its *Modeler*, which adopts the basic correctness principles and verification procedures

we developed in ADEPT [4]. Additionally, it provides features for partitioning global processes into several process fragments, for allocating the resulting process fragments to different machines (e.g., mobiles devices), and for linking process activities with activity components (cf. Sect. 4.4). Consider again our example from Section 2. Using *MARPLE Modeler*, for instance, the process fragment, coordinating data collection steps and running on the mobile device of the patient, can be defined (see Fig. 5). The *MARPLE Modeler* is subdivided into three areas: The toolbar on the left side ⓐ depicts available modeling elements: `Activity Nodes` and `Connectors`, `Data Elements`, and `Activity Templates`. ⓑ depicts the main area of the *Modeler*: Basically, the process model can be created through drag & drop operations, which copy elements from the toolbar to the design area. Finally, the top toolbar ⓒ provides useful modeling functions (e.g. for layouting the models). The *MARPLE Modeler* is based on the *Windows Presentation Foundation* as part of the *Microsoft .NET Framework*. The process elements depicted within area ⓑ are defined using XML. The editor stores modeled process fragments via XSLT-transformation according to the internal format required by the *MARPLE Repository*.

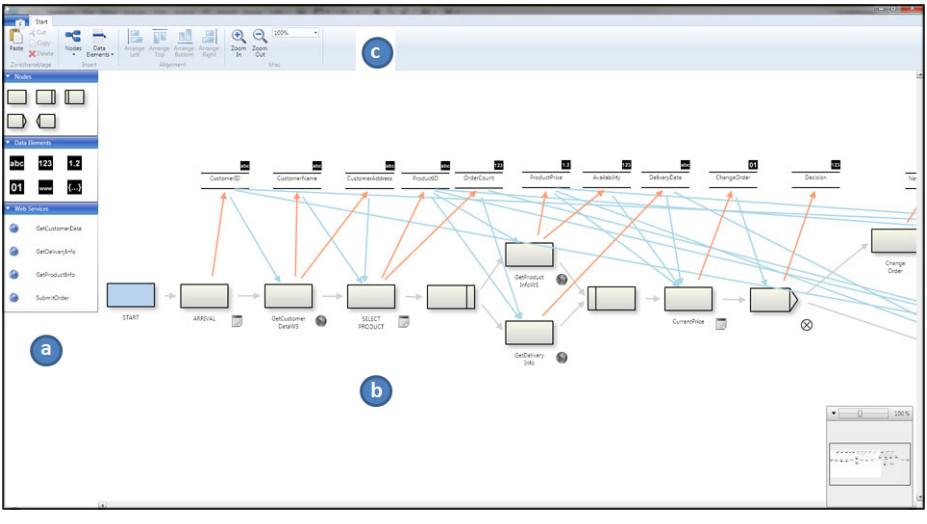

Fig. 5. MARPLE Modeler

Configurable user forms: User-friendliness is one major aspect for mobile applications. As we learned in our healthcare case studies, physicians and nurses rapidly become discouraged when being confronted with inappropriate user interfaces. We therefore integrated a form editor with the *MARPLE Modeler*. It comprises features to cope with the problem of limited screen size on mobile devices (e.g., if too many data elements shall be displayed in one form, this can be splitted into two forms). In addition, to meet context demands (like in emergency situations), we added specific features for enabling more sophisticated

user interactions. For example, instead of entering data manually, a physician or nurse may enter it only through pre-defined controls.

4.3 MARPLE Mobile Engine

When developing the *MARPLE Mobile Process Engine*, we re-used basic concepts and design principles of the ADEPT process management technology, which we had developed during the last decade [3]. In particular, we adopted the ADEPT process meta model, applied its fundamental correctness notions as well as verification procedures, and support flexible and adaptive process enactment on the mobile device as well. The latter enables dynamic structural adaptations of process fragment instances running on the mobile device; e.g., to cope with contextual changes in the environment or exceptional situations. Basic to the support of such ad-hoc changes is the *MARPLE Mobile Deviation Service*. Despite these commonalities with ADEPT it is noteworthy that we provide a complete new implementation of the kernel of the *MARPLE Mobile Process Engine* in order to meet performance requirements of mobile scenarios and to cope with issues specific to mobile processes (e.g., broken connections and limited GUIs). In particular, the implementation framework on which MARPLE is based differs from the one used in ADEPT - ADEPT relies on JAVA, while MARPLE is based on *.NET Compact Framework*. In the following we describe selected services of the *MARPLE Mobile Process Engine* in more detail.

Configuring mobile devices: When a mobile device is added to the *MARPLE* environment, it first needs to be equipped with basic software services. Amongst others, this includes *Core Communication Services (CCS)* as basic pillar of the *MARPLE Mobile Process Engine V1.5*. Thereby, we apply a light-weight approach; i.e., services initially not needed are not uploaded to the device. Following this, the mobile device can connect to the *MARPLE Mediation Center*. When starting the *MARPLE* configuration procedure on the mobile device using the *MARPLE Mediation Center*, CCS dynamically loads the *MARPLE Mobile Process Engine*, the *XML Persistence Manager*, and selected *process* as well as *activity templates* (see also Section 4.4) to this device. The *XML Persistence Manager* fulfills two functions: First, it allows (de)serializing process models in order to exchange them between the *MARPLE Mediation Center* and the mobile devices. Second, it enables the storage of process models on mobile devices.

Enactment of processes on mobile devices: MARPLE allows to deploy process models and fragments, respectively, to mobile devices and to create corresponding instances. In the first version of the *MARPLE Mobile Process Engine* we only considered the execution patterns *sequence* and *conditional* routing. Later we added support for the *parallel execution* of process activities on mobile devices, which is required for performing "background"

Fig. 6. Parallel Execution

activities, e.g., database operations or web service calls (cf. Fig. 6). Parallel processing of activities on mobile devices can be also utilized to reduce throughput times and error probabilities as well as to enable better monitoring. Due to limited screen size of mobile devices, however, concurrent execution of activities on mobile devices should be more restrictive than on normal process servers. Displaying multi-windows simultaneously, for example, is not meaningful. For process fragments running on mobile devices, *MARPLE* therefore disallows the concurrent processing of forms referring to different parallel branches. As example consider Step ⑥ of our healthcare scenario from Section 2. Here messages are sent to the patient device in parallel to the form-based process step *Prepare Patient* (as performed by homecare). Parallel processing is very helpful in the given context to foster monitoring. Let us consider that the process instance of the patient shall be monitored by a clinical doctor and homecare is involved. The homecare process fragment sends (cf. Fig. 2) in parallel information back to the activity *Prepare Patient*. This enables the clinician to already monitor the activity *Prepare Patient* during execution without waiting of its finishing. Especially, if the activity has a long duration, this information could be very important.

Fig. 7. Ad-hoc Deviations

Ad-hoc deviations during mobile process enactment: The ability to deviate from pre-modeled process templates during runtime is crucial for mobile process support. Both changes in the local environment (e.g., blood pressure of a patient increases) and changes in the infrastructure or global environment (e.g., a particular sensor is not measuring data) might require ad-hoc changes of a mobile process. For example, if results are missing during the examination of a patient, but are urgently needed, medical tests or procedures may have to be dynamically added. To be able to adjust a particular instance of the mobile process correctly and quickly during runtime, we implemented a sub-component of the *MARPLE Mobile Process Engine* that enables users to locally adapt processes running on the mobile device. Thereby, *MARPLE* displays the process to the user and offers a number of context-enabled changes (cf. Fig. 7). The offered set is based on context information provided locally through the *MARPLE Mobile Process Engine* (e.g., regarding the current status of the executed fragment and the role of the respective user). Change operations implemented so far include the insertion and deletion of single activities. In the current implementation only human tasks (i.e., form-based activities) can be added or deleted locally. In our future research, remote adaptation of process fragments running on mobile devices will become a topic of interest as well.

Integration with calendar systems: On the one hand many companies use collaboration tools like e-mail clients and calendar systems with built-in schedule capabilities. On the other hand, mobile scenarios like the one from Section 2 are

usually linked with many related time events, of which some can be related to calendar items. Regarding our healthcare scenario, one important time event at the clinic concerns the triggering of the whole process. Assume that doctors have to trigger such events manually. Then it would be highly welcomed by them if the activity had been linked to their personal calendar. Amongst others this would reduce omission errors. Furthermore, if the activity has to be postponed, the infrastructure can explicitly cope with this situation. We therefore interlinked the *MARPLE Architecture* with *Microsoft Outlook*. With the help of the *MARPLE Mediation Center* the interaction between the two tools can be configured. The *MARPLE Mobile Process Engine* then starts process instances on the mobile device accordingly to pre-defined calendar items.

4.4 Activity Templates

In *MARPLE*, single process steps can be implemented based on reusable *Activity Templates* (cf. Fig. 8). These encapsulate pre-manufactured application components.

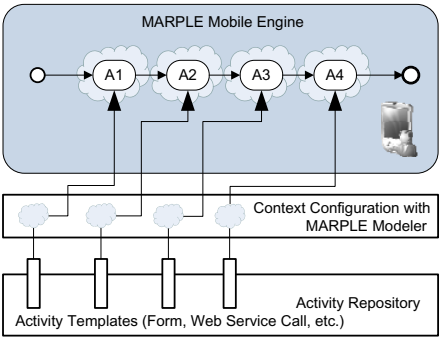

Fig. 8. Activity Templates

Their implementation must be a *dll* component of the *.NET Compact Framework*. We store available *activity templates* in the *MARPLE Repository*. Based on the latter *activity templates* are linked to process steps at design time using the *MARPLE Modeler*. Thereby, they have to be configured to the given context. *Activity Templates* further must be provided with a number of configure parameters depending on the given type (e.g., concrete parameters for a web service call like the message format or URI). During process execution, the *MARPLE Mobile Process Engine* loads the *activity templates* incrementally when activating process steps (cf. Fig. 8).

As standard activity templates MARPLE provides support for invoking user forms and web services as well as for evaluating transition (xor) conditions. Based on respective core activity templates, simple mobile scenarios like the one from Section 2 can be implemented. As example consider activity ⑪ in our scenario, where the user interacts with a form to enter his blood pressure value.

From other mobile process scenarios we implemented with *MARPLE*, we revealed the need for additional kinds of activity templates covering locations awareness, database connectivity, and barcode management.

Database activity template: When investigating mobile scenarios from the sales domain we started considering database functions for mobile process

management. We talked to sales representatives from the insurance domain and collected characteristic requirements. Most prominently, they claimed that data management in respect to their mobile device and applications is not satisfactory. An appropriate data management component needs to cover three aspects. First, required customer data should be automatically transferred to the mobile device in advance (e.g., by utilizing information about customer appointments). Second, for a process running on the mobile device and its interlinked processes in the backend, data consistency must be ensured. Third, if no connection can be made during the meeting with a customer, correct re-transfer of data into the backend system should be provided. All these steps require a database connection in order to perform respective queries. To meet this requirement, we implemented a database activity template. Since parallel processing of activities on the mobile device is possible in *MARPLE* any activity based on the database template can frontload data for later process activities. For example, while the sales manager runs form-based activities for entering customer data, certain data management steps can be executed in parallel. Additionally, data collected via the mobile device can be transferred back to the backend system at the earliest point in time. Consequently, this data can be used for recovery purposes, e.g., when an abnormal situation (e.g., broken device) occurs.

Fig. 9. Barcode Reading

Activity templates enabling sensor awareness: One major goal of *MARPLE* is to provide personalized and situation-aware mobile processes. For reaching this objective the integration of real-time data as provided by sensors (e.g., blood pressure of a particular patient can be gathered via *Bluetooth*) with mobile processes is needed. Our healthcare scenarios indicated that often patient data can be gathered via sensors (e.g., measuring pulse, respiration, blood pressure, and body temperature). The gathered data then has to be processed, aggregated, and interpreted considering the given context. Due to lack of process-awareness, effective processing of this data is usually limited. We further learned from our clinical scenarios that sensor-gathered data is usually only visualized based on specialized applications provided by the sensor vendor. In order to utilize such context data (e.g., a medical parameter has exceeded its threshold multiple times) we are developing *Activity Templates* that enable sensor-awareness. We have already implemented a *Barcode Activity Template*, which enables users to scan data collectable from packages (e.g., drug packages) (cf. Fig. 9). As example consider the clinical admission of patients. The barcode of medications brought along by the patients to the clinic could be scanned and archived in the patient record. In the context of another project in the field of airline catering, we are currently working on a *RFID Activity Template*. This template shall enable users to read RFID tags and utilize gathered data within the mobile processes respectively.

Map Activity Template: Mobility implies a larger extent of flexibility on the one hand, and a lot more sources of problems (e.g. no network connection, limited battery life) on the other hand. This must be considered in the context of mobile process execution as well. Location awareness is one feature we should add when compared to classical workflows. *MARPLE* therefore supports a *location activity template*, which can display either the current user location, if a GPS module exists, or any location based on given coordinates (cf. Fig. 10). Regarding our sample process this activity template can be used to realize Step ⑤ in order to guide the homecare service when visiting the patient. Furthermore, context information about location can be used to provide context-specific assistance for users (e.g., by only displaying those instances to them which are relevant in the given context). Additionally, coordinates of the mobile devices can be periodically submitted to the *Mediation Center*. This way resources (e.g. workload, errands) can be managed in a more efficient way.

Fig. 10. Street Map Integration

4.5 Summary of the MARPLE Features

Table 2 summarizes core features of MARPLE components:

Table 2. Marple Architecture Feature Overview

Component	Services	Description
Mediation Center	Design	Modeling processes via "drag&drop"-operations and customization facilities.
	Control	Allocate processes on mobile devices and remotely monitor running instances. Perform instance migrations to other devices or ad-hoc deviations if needed.
	Repository	Save templates, process instances, and configurations.
	Maintenance	Manage mobile devices and configurations.
Mobile Engine	Process Enactment	Execute processes on a mobile device. Supported activity templates are: forms, web services, database access, barcode scanning & processing, and location services.
	Process Deviations and Visualization	User interface for inserting new activities and for skipping existing ones.
	Persistency	Persists current instance data on mobile device and additionally stores the process template as well as activity templates on the mobile device.
	Communication	Built-in server for handling the overall communication with the *Mediation Center*.
Integration with Calendar System (Outlook)	Scheduling	Assign appointments to process steps, i.e., link process instances with calendar items.

5 Realizing the Healthcare Scenario with MARPLE

We revisit our scenario from Section 2 and show how it can be realized using MARPLE. Fig. 11 depicts the user interface of the *MARPLE Mediation Center*

and shows a concrete process instance running on a mobile device as it can be monitored with the *MARPLE Mediation Center*. Note that this perspective displays both the current status of the mobile process and the data values collected during process execution (see ⑦). Obviously, this is exactly the information a medical professional needs when remotely monitoring the patient process.

How does the patient process fragment look like in *MARPLE*: Fig. 11 depicts a part of this model together with instance-specific markings. Activity ② constitutes a *receive* activity which is waiting for an incoming message requesting a health check. The following three activities constitute data collection steps, which are either implemented as user forms or sensing activities. The blood pressure, in turn, is gathered via a bluetooth activity template which is linked with the blood pressure measurement machine. Blood glucose and ECG recordings are entered by users via form-based activities; i.e., the user of the mobile device gets respective requests in his worklist and then has to fill in the two forms (e.g. see the PDA display in Fig. 4).

Following data collection, activity ③ is automatically executed. It invokes a web service at the clinic to report about measured results (e.g., to add them to the electronic patient record). The subsequent activity ④ then waits until a message is received either from the clinic or from homecare. The administrator toolbar on the left side of Fig. 11 ⑧ displays available functions for managing process templates, users, mobile devices, and mobile device settings. Further, ⑥ displays the list of currently released process templates, which can be assigned to registered mobile devices.

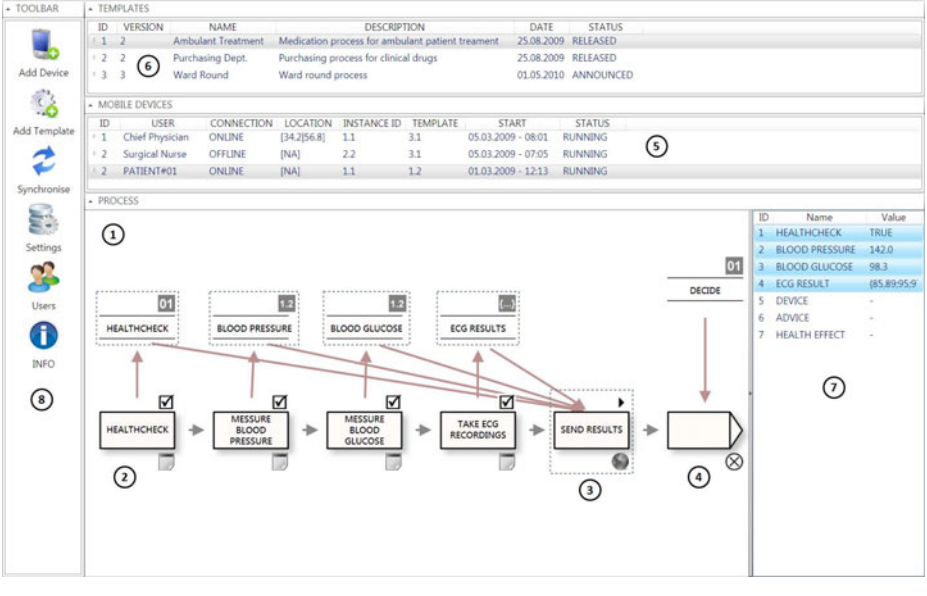

Fig. 11. MARPLE: Mediation Center

6 Evaluation of Mobile Process Support with MARPLE

We applied MARPLE to process scenarios from healthcare, logistics and sales, which we implemented using the *MARPLE Architecture*. We learned that the provision of a light-weight process engine for mobile devices is essential in order to realize human-centric processes. Furthermore, our prototypical implementations revealed additional demands we picked up in this paper. Table 3 gives an overview of the considered scenarios and relates them to *MARPLE* features.

Table 3. Marple Features Usage

MARPLE Features / Scenarios		Scenario 1 *Medical Telecare*	Scenario 2 *Clinical Ward Round*	Scenario 3 *Ambulance Service*	Scenario 4 *Insurance Sales (CRM)*	Scenario 5 *Airline Catering*
Activity Templates	Barcode	o	x	o	-	x
	Database	o	x	-	f	x
	Map	o	-	x	x	f
	User Form	x	x	x	x	f
	Web Service	o	x	o	f	x
Calendar Integration		f	o	-	o	f
Configurable User Forms		f	f	f	o	f
Local Process Deviations		o	f	o	o	f
Parallel Activities		o	x	f	f	x
Process Migration		o	f	o	o	o
Process Monitoring		x	o	o	o	x

(-) : not needed **(o)** : rarely needed **(f)** : frequently needed **(x)** : heavily needed

7 Related Work

In literature, we can find several approaches which focus on logical models for mobile processes on the one hand and approaches addressing architectural and implementation issues of light-weight process engines on the other hand. Regarding the first category, for example, several approaches exist for the partitioning of BPEL processes [5,6,7,8]. A similar approach has been suggested in the context of $ADEPT_{distribution}$ [9,10]. However, none of the two approaches has provided an architecture for mobile process support as MARPLE does. Taking network dynamics as core demand for mobile process engines, several approaches deal with failures and exceptions like broken connections or lack of communication facilities [11,12,13,14]. Respective tools usually apply web service standards and base process execution on BPEL or more specific execution models derived from it. We consider the use of BPEL as process execution language as too low level, particularly if it shall be possible to dynamically evolve or adapt mobile processes during runtime. Instead we provide a process model at a higher level of abstraction that can be adapted by both remote users and users of the mobile device. We further believe that self-healing techniques and migration management will be crucial for mobile processes in the large scale. However, existing approaches dealing with these aspects in process-aware information systems, again focus on

BPEL as execution language [6,15]. The same applies in respect to mobile processes [16]. As described, combining real-time data and domain knowledge raises additional challenges for mobile process management. The MobiHealth project [17], for example, supports context-aware services which allow to integrate sensor data. However, context-awareness is restricted in the sense that execution can only be switched between pre-configured process variants when the context is changing. Generally, approaches allowing to switch between pre-configured process variants can be found in other projects as well [18,16]. However, the fusion of real-time data and domain knowledge with mobile processes has not been considered in a suitable manner yet. Finally, there are approaches focusing on process-aware systems using web services and running on mobile devices [19,7,20]. They enable simple web service flows on mobile devices based on a mobile engine. Approaches only focusing on web service calls, however, restrict the potential of mobile processes in several respects.

8 Summary and Outlook

We introduced the MARPLE approach and described how its core components enable the execution and monitoring of processes (fragments) on mobile devices. Our overall vision is to provide sophisticated mobile process support; i.e., to realize generic process management features including support for process instance changes, process instance migrations, sensor data integration, etc. To foster this vision we base our presented work on core design principles and fundamental concepts we developed in our ADEPT project as well as our $ADEPT_{distribution}$ project [10,21]. In future work we will extend the *MARPLE Modeler* such that it provides sophisticated methods for modeling complex process choreographies including numerous process servers, devices, and actors possibly distributed over many departments. This will include, for example, a methodology for correctly partitioning processes models, for allocating resulting fragments on different machines and devices based on more intelligent allocation techniques, and for synchronizing different process fragments at runtime. In particular, we will adopt and extend concepts from autonomic computing and self-healing systems [22] to cope with the many failure scenarios in connection with distributed and mobile applications. This will be crucial when realizing mobile processes in the large scale.

References

1. Smith, H., Fingar, P.: Business Process Management: The Third Wave. Meghan-Kiffer Press (2003)
2. Bonnici, E., Welch, P.H.: Mobile processes, mobile channels and dynamic systems. In: IEEE Congress on Evolutionary Computation (CEC 2009), pp. 232–239 (2009)
3. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development 23, 81–97 (2009)
4. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 115–135. Springer, Heidelberg (2009)

5. Baresi, L., Maurino, A., Modafferi, S.: Workflow partitioning in mobile information systems. In: MOBIS 2004, pp. 93–106 (2004)
6. Baresi, L., Guinea, S.: Dynamo and self-healing BPEL compositions. In: Proc. 29th Int. Conf. on Software Engineering, pp. 69–70 (2007)
7. Hahn, K., Schweppe, H.: Exploring transactional service properties for mobile service composition. In: MMS 2009, pp. 39–52 (2009)
8. Schmidt, H., Hauck, F.J.: SAMPROC: middleware for self-adaptive mobile processes in heterogeneous ubiquitous environments. In: Proc. 4th Middleware Doctoral Symposium, New York, NY, USA, pp. 1–6 (2007)
9. Bauer, T., Reichert, M., Dadam, P.: Intra-subnet load balancing in distributed workflow management systems. Int'l Journal of Cooperative Information Systems 12, 205–323 (2003)
10. Reichert, M., Bauer, T.: Supporting ad-hoc changes in distributed workflow management systems. In: Meersman, R., Tari, Z. (eds.) CoopIS 2007. LNCS, vol. 4803, pp. 150–168. Springer, Heidelberg (2007)
11. Kunze, C.P.: DEMAC: A distributed environment for mobility-aware computing. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 115–121. Springer, Heidelberg (2005)
12. Hackmann, G., Haitjema, M., Gill, 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)
13. Schmidt, H., Kapitza, R., Hauck, F.J.: Mobile-process-based ubiquitous computing platform: a blueprint. In: Proc. 1st Workshop on Middleware-Application Interaction, pp. 25–30 (2007)
14. Gerhard, S., Jürgen, M., Erich, S.: Building a modular service oriented workflow engine. In: Int. Conf. on Service-Oriented Comp. and Applications (2009)
15. Baresi, L., Guinea, S., Pasquale, L.: Self-healing BPEL processes with DYNAMO and the JBoss rule engine. In: Int. Workshop on Engineering of Software Services for Pervasive Environments, pp. 11–20 (2007)
16. Zaplata, S., Kottke, K., Meiners, M., Lamersdorf, W.: Towards runtime migration of WS-BPEL processes. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 477–487. Springer, Heidelberg (2010)
17. 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.: Mobihealth: mobile services for health professionals. Technical Report TR-CTIT-06-38, Enschede (2006)
18. Hallerbach, A., Bauer, T., Reichert, M.: Capturing Variability in Business Process Models: The Provop Approach. Journal of Software Maintenance and Evolution: Research and Practice 22(6-7), 519–546 (2010)
19. Battista, D., de Leoni, M., De Gaetanis, A., Mecella, M., Pezzullo, A., Russo, A., Saponaro, C.: ROME4EU: A web service-based process-aware system for smart devices. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 726–727. Springer, Heidelberg (2008)
20. Zaplata, S., Dreiling, V., Lamersdorf, W.: Realizing mobile web services for dynamic applications. In: Godart, C., Norbert Gronau, S.S.G.C. (eds.) I3E 2009, pp. 240–254. Springer, Heidelberg (2009)
21. Reichert, M., Bauer, T., Dadam, P.: Flexibility for distributed workflows. In: Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments, pp. 137–171. IGI Global, Hershey (2009)
22. Gui, C., Mohapatra, P.: SHORT: Self-healing and optimizing routing techniques for mobile ad hoc networks. In: Proc. 4th ACM Int. Symposium on Mobile Ad-hoc Networking & Computing, pp. 279–290 (2003)