Context-based Cooperation in Mobile Business Environments – Managing the Distributed Execution of Mobile Processes

Current mobile business tasks are, in most cases, restricted by the resources of their respective mobile execution environments and thus still relatively simple. However, even more complex and structured business applications could be realized in more flexible ways by delegating (sub-) tasks to other systems. For this purpose, the concept of mobile processes enables a context-based cooperation of mobile and stationary devices, users and services. It allows running process instances to migrate in local vicinities and, thereby, increases the execution probability of complex tasks significantly. A self-organized and context-aware distribution strategy enables economic and efficient execution of even ad hoc tasks. Finally, reliable business applications are realized by additional mechanisms from the areas of process management, transaction handling and secure messaging.

DOI 10.1007/s12599-009-0060-5

The Authors

Dipl.-Inf. Sonja Zaplata
Prof. Dr. Winfried Lamersdorf
Universität Hamburg
Department Informatik
Vogt-Kölln-Straße 30
22527 Hamburg
Germany
{zaplata | lamersdorf}@
informatik.uni-hamburg.de

Dr. Christian P. Kunze
Steria Mummert Consulting AG
Hans-Henny-Jahnn-Weg 29
22085 Hamburg
Germany
christian.kunze@steria-mummert.de

Received: 2008-09-29
Accepted: 2009-02-15
Accepted after one revision
by Prof. Dr. Buhl.

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Zaplata S, Kunze CP, Lamersdorf W (2009) Kontextbasierte Kooperation für mobile Geschäftsanwendungen – Dezentrale Ausführung und Management von mobilen Prozessen. WIRTSCHAFTSINFORMATIK. doi: 10.1007/11576-009-0183-9.

1 Motivation

In the modern services society, mobility has become one of the main driving factors. Supported by the decentralization trends in information and communication technology, a great diversity of portable and networked devices has evolved. The use of these devices has increased continuously and has lead to a rather ubiquitous availability of information and services for mobile users (cp. also Fleisch and Dierkes 2003, pp. 611–620). Despite that, even contemporary mobile applications are still restricted to the capabilities of the executing device. For instance, mobile devices equipped with communication technologies such as Bluetooth or Infrared are normally unable to access the Internet or to integrate Web Services. On the other hand, many local services cannot be accessed by stationary devices due to their immobility. In consequence, the complexity of current application tasks is often limited to the initiating device's capabilities and its direct local environment. Resources which are accessible from other devices only via mobile networks therefore usually remain unusable for the dynamic adjustment of mobile applications.

However, there is a wide range of applications which would benefit from a more adaptable, self-organized infrastructure which is also able to support more complex and even spontaneous tasks arising in mobile environments and involving several mobile and stationary devices. Some interesting application areas for such tasks are briefly outlined in the following distributed collaboration scenarios:

- *Collaborative transport chain control*
  In current transport logistics scenarios, freight traffic is often monitored by wireless sensor technology (e. g. RFID), which is mostly passive and statically bound to the transport container. Thus, sensitive goods can be tracked from remote and can be controlled temporarily whenever special-purpose devices are available, e. g. at the shipping area or the container terminal. A logical next step in the evolution of such systems is to allow a self-acting behavior and to adapt this behavior to the individual characteristics of the goods (for example to measure the temperature in a freezer container and to send alarms in case of irregularities). To avoid the need for buying new hardware for any such purposes, the collaboration of already existing devices and resources in the mobile vicinity is an attractive alternative. Nevertheless, a typical transport chain generally involves several locations and the occurrence of events usually cannot be foreseen. An appropriate software system therefore should detect the potential of its vicinity to be able to adapt to its local conditions dynamically.

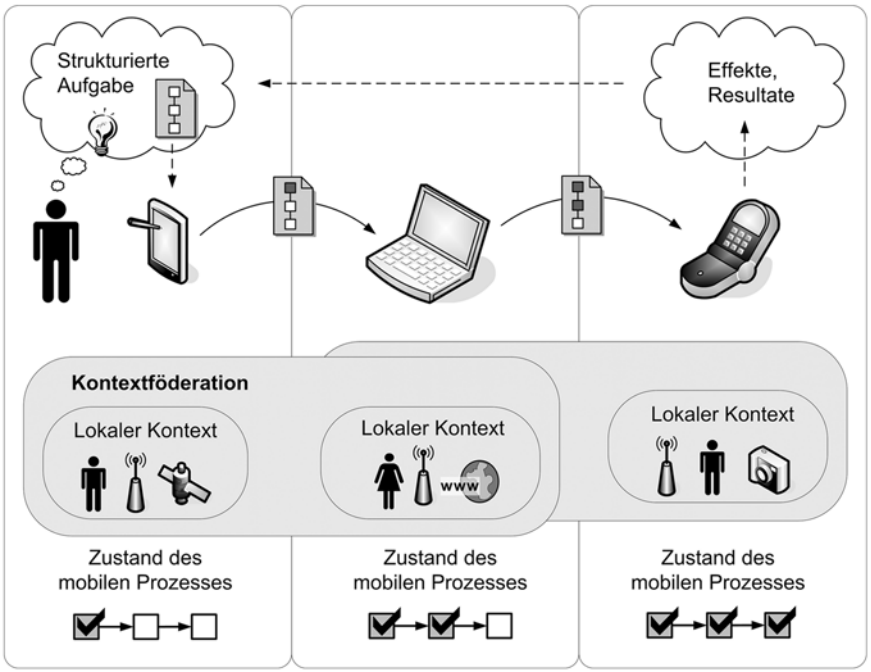

Fig. 1 Context-based cooperation for mobile processes

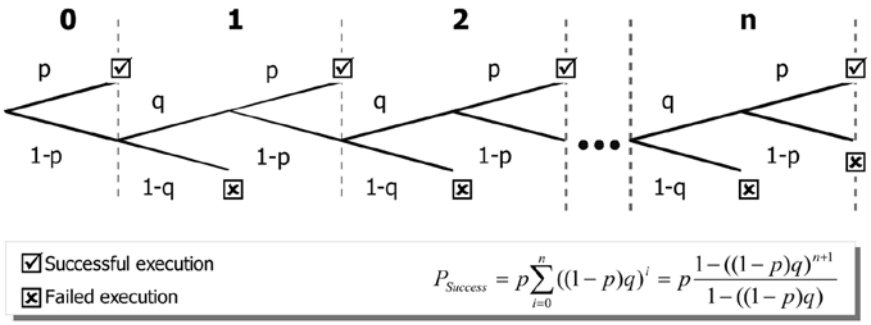

Fig. 2 Probability tree of successful mobile process execution (Kunze et al. 2008, p. 61)

- *Health care management*
  Another use case can be found as part of disease management programs. In such an environment, more and more patients carry mobile devices, including equipments for medical observation, instruments to remind them of taking medication as well as traditional cell phones, e. g. to request emergency aid. Both local doctors as well as hospitals collect and process patients' medical records electronically and exchange data in a standardized manner. However, in order to enable individual applications for each patient across multiple of these systems, more sophisticated forms of collaboration between formerly individual systems are required. This could enable, e. g., applications to monitor the patient's health, call the most appropriate medical doctor if necessary, and integrate medical service administration, e. g. accounting and/or insurance functions, automatically.
- *Decentralized mobile workflow processing*
  A third scenario could be imagined in the area of business service management where new structured tasks (e. g. orders of costumers) appear spontaneously and have to be processed by a mobile work team. In such a scenario, mobile workers are equipped with mobile devices, such as personal digital assistants or mobile phones which usually have only restricted access to customers' computing systems. Without a central infrastructure, a supporting workflow system has to organize the execution of the task autonomously and is itself responsible for allocating all required human resources and automated services in order to execute the task in a distributed way.

In summary, the execution of tasks consisting of several sub-steps could be enabled or enhanced by cooperation between different devices with complementary capabilities. In this contribution, the technical representation of an executable structured application task is called a *process*. Such a process involves, in general, services according to a Service-Oriented Architecture (SOA) (cp. Papazoglou 2003, pp. 3–12) as well as manual or interactive human (sub-) tasks which are specified in an operational and executable way. Nevertheless, the operational description of tasks can be used to (totally or partially) implement business processes if these can be mapped to concrete functionalities provided by the service network.

In order to propose technical support for such cooperative application task scenarios, this contribution proceeds as follows: It first summarizes our previous research in the corresponding area – mainly based on a deductive methodology. Initially, the basic idea of context-based cooperation and its corresponding hypothesis of an increasing execution probability are presented. Respective requirements for a supporting middleware infrastructure and related approaches in the area of business process management and mobile computing are analyzed in the following section. Concluding part one, an adequate concept for the technical realization of mobile processes is elaborated. The second part of this contribution analyzes the applicability of the developed concepts in the area of practical business applications. Accordingly, relevant challenges are identified and classified into issues concerning process management, transaction processing, and security. Based on the approach developed so far, exemplary concepts and solutions for these three main challenges are pointed out. Finally, it is shown how the resulting overall concept of mobile processes can be implemented and tested against the specified hypothesis, using both experimental prototype evaluations as well as a scenario-based approach.

2 Concept of context-based cooperation

As presented above, the execution of mobile applications is often limited by

Deutsche Telekom Laboratories

We shape the future

Deutsche Telekom Laboratories is Deutsche Telekom's research and development institute based in Berlin. It is simultaneously a scientific institute organized under private law and associated with the Technische Universität (TU) Berlin. At Deutsche Telekom Laboratories, scientists from across the globe work together with experts from the Group to develop new services and solutions for Deutsche Telekom's customers. Establishing new companies (spin-offs) is another method for making use of research output.

Cooperation with the TU Berlin, other universities and industry partners creates a bridge between business and science in order to turn ideas into marketable innovations as quickly as possible. As part of this, Deutsche Telekom Laboratories focuses on five fields of innovation (5 i):

- **I**ntuitive Usability of services and devices
- **I**ntegrated Service Components
- **I**ntelligent Access
- **I**nfrastructure for IT and telecommunications
- **I**nherent Security

The business and information systems engineering offers useful interdisciplinary approaches for all these areas of innovation. Subject matter includes, for example, modeling, methods and tools for process innovations, agile architectures for information and communication technologies (ICT), technology-oriented management approaches and techno-economic assessments. The aim is to safeguard the economic sustainability of innovations for the Group.

Deutsche Telekom Laboratories is divided into two areas: The Innovation Development Laboratory focuses on market-centric research and development within a timeframe of up to three years.

The basic and technology research of the Strategic Research Laboratory has a long-term focus. Common goal: Deutsche Telekom Laboratories is looking to become one of the world's leading research and development institutions in the field of new ICT.

An institute was set up together with Ben-Gurion University in Beer Sheva, Israel, in 2006. Since 2008, Deutsche Telekom Laboratories has also been represented in Darmstadt. Another project office was opened in the Silicon Valley, United States, in January 2009.

Contact:
Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7
10587 Berlin, Germany
E-mail: wi.laboratories@telekom.de
www.laboratories.telekom.com

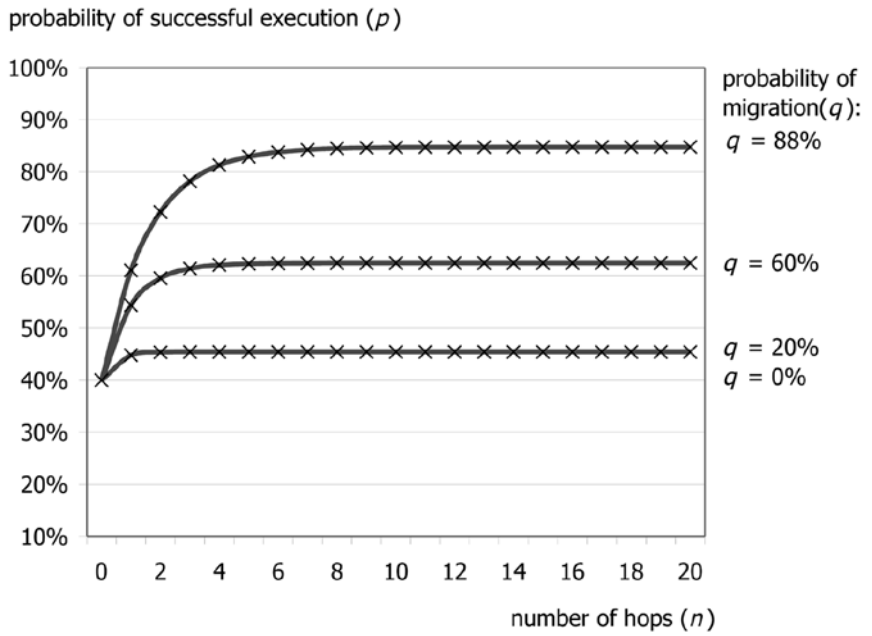

Fig. 3 Execution probability of process migration variants ($p = 40$ %)

the capabilities of the respective executing device. Compared with stationary devices, such limitations are intrinsic characteristics of mobile computing systems, because they directly result from the mobility of users, devices, and applications (Satyanarayanan 1996, pp. 1–7). This means that even despite of the ongoing evolution of mobile systems they have relatively light computing power, memory resources, and no reliable energy supply – leading to the fact that they can often execute only a subset of complex application tasks. Also the continuous increase of capabilities and connectivity leads to novel, more specialized, and even smaller devices with new requirements and challenges (cp. Adelstein et al. 2005, pp. 191–199 and 203–208). As often not all required resources or services can be accessed locally, many types of structured applications and tasks spanning several devices cannot be executed or are likely to fail due to limited resources of the initiating device. In order to also make use of other services and resources in the mobile environment, a possible solution is to forward the upcoming task to other participants. Therefore, the task should preferably be specified in a structured and operational, but technology-independent way. Furthermore, the application-independent acquisition of context information (e. g. reachable devices and their resources) is an important prerequisite to realize the assignment of such spontaneously arising tasks.

These basic observations lead to the concept of mobile processes as a representation of such structured abstract application tasks – including the use of context knowledge to execute service compositions by integrating the capabilities of different mobile nodes of a local environment. As these processes enable cooperation among all participating (mobile and other) systems, applications implemented as mobile processes are able to realize a form of context-based cooperation. A mobile process can be defined as a goal-oriented composition of arbitrary services and manual tasks which may span several heterogeneous mobile and stationary devices, users, and services. In order to (potentially) use all the capabilities and resources provided in its environment, the process can migrate to other devices, for instance to share the functionality provided by these nodes. Three different scenarios can be distinguished here: First, a task can be executed locally by an application running on the same (mobile) device. Second, if such local application is unavailable, the device can search for adequate services on other devices in its local environment to execute the task remotely. Third, in case the direct vicinity does not provide the required resources, the task's description can be transferred to a remote device in order to enable the execution in a different context.

Fig. 1 shows a mobile process migrating between three devices depending on the discovered context. As long as the process engine of a device is able to bind local or remote services to the current activities, it is responsible for the mobile process. In cases of failures or lack of suitable service instances the engine has to find other devices to transfer the remaining process and its execution to one of them. In order to ensure the initiator's goals even on foreign devices, additional non-functional requirements can optionally restrict participating services, users, and devices to specified quality characteristics. Furthermore, the mobile process does not necessarily have to return to its initiator since, e. g., the initiator may only be interested in the effects of the execution as depicted exemplary in **Fig. 1**.

Mobile processes and the procedure of process migration open up new execution contexts including other and, maybe, more suitable devices as determined by the heterogeneity of the vicinity. **Fig. 2** depicts this procedure using p to denote the probability of a single device being capable of executing the current task (locally or remotely), q denoting the probability of process migration and n representing the number of hops caused by the migration.

The successful execution probability for a migrating process can be calculated as a converging geometric series of the likelihood of successful sub-task execution anywhere in the mobile vicinity (cp. **Fig. 2**). Some exemplary values are presented in **Fig. 3**, showing the probabilities of successful process execution with exemplary migration probabilities of $q = 0$ %, $q = 20$ % $q = 60$ % and $q = 88$ %, while p is assumed to be constantly equal to 40 %. As to see, the estimated probability of a successful execution increases considerably already after a few hops, especially if there is a high heterogeneity and thus a high migration probability.

In summary, the probability of successful process execution – involving arbitrary tasks and heterogeneous devices – is influenced by the migration which is itself dependent on the potential to detect and integrate heterogeneous context dynamically. However, such different context cannot be determined in advance. The context model and management system of a supporting middleware infrastructure therefore have to be generic and application-independent, even at runtime – a system software characteristic which is hardly supported by traditional context-

aware middleware systems. Furthermore, the process itself and its execution environment must support the migration of running process instances to other systems. The following section presents basic requirements for such a generic context-aware middleware approach for mobile processes and compares them with other existing and related approaches.

3 Requirements analysis and review of related approaches

In order to describe processes in ways which allow execution strategies as described above, mobile process representations have rather similar requirements as traditional (business) processes. These are among others the need for the ability to express the business logic with its data and control flow, the participating parties (as roles or individuals), and routines to recover from failures. However, there are also some additional requirements based on the characteristics of mobile environments and their distributed execution strategy. For example, mobile process descriptions must be lean and simple to save memory, computing power, and energy resources. They also have to include mechanisms to handle communication failures and for the distribution of the process itself. This means in particular that the state of the process and the initiator's non-functional requirements for the execution of the process must be expressible. Due to the dynamism, the processes also need a *late binding mechanism* to assign not only service instances but also service technology at runtime. Therefore, the process description must support an abstract notation of the desired services (Kunze et al. 2006, p. 4).

In order to find an adequate basis to describe the highly dynamic processes which may run on mobile distributed computing systems, a number of existing process description languages can be considered. The *Business Process Execution Language for Web Services (WS-BPEL)* (Barreto et al. 2007), the *Web Service Choreography Interface (WSCI)* (Arkin et al. 2002), ebXML's *Business Process Specification Schema (ebBPSS)* (Riemer 2001), the *XML Process Definition Language (XPDL)* (WfMC 2005), and the *Java Process Definition Language (JPDL)* (JBoss 2005) have been evaluated with respect to a catalogue of 19 identified requirements (cp. **Tab. 1**).

Tab. 1 Requirements catalogue and analysis of existing process description languages (Kunze et al. 2006, pp. 5–36)

	WS-BPEL	WSCI	EbBPSS	JPDL	XPDL
Service composition	+	+	+	+	+
Interactive and manual tasks	–	–	–	+	+
Control flow	+	+	+	+	+
Data flow	+	+	+	+	+
Participants (users, roles)	+	+	+	+	+
Specification of non-functional requirements	–	+	+	–	–
Late binding / service discovery	+/–	+	+	+	+
Distributed administration	–	+/–	–	–	–
Error handling	+	+	–	–	–
Scalability	+	–	–	+	+
Extensibility for user-defined elements	+	+	+	+	+
Security support	–	–	+	–	–
Transaction support	+	+	–	–	–
Audit trail	–	–	+	–	+
Technology-independence	–	–	–	–	+
Priority management	–	–	–	–	+
Economic consumption of memory / computing power	–	+	–	+/–	+/–
Avoidance of communication overhead	+	+	+	+/–	+/–
Connection management	–	–	–	–	–

+: applicable +/-: in parts -: not applicable

Most importantly, the catalogue covers the mentioned requirements for distribution, dynamism, flexibility, error handling, and technology-independence. Several of the presented languages allow late binding of process participants or integrate concepts to choose participants by their respective quality. However, none of them supports the transfer of running process instances or allows for a completely distributed administration and execution of (mobile) processes. Additionally, concepts for handling connection resets and for saving resources (e. g. memory, computing power) have not been considered yet, since these process description languages have been developed mainly for reliable centralized workflow engines. As the most critical point, except for XPDL all of the mentioned description languages focus on a special technology or description format, e. g. WS-BPEL is restricted to compose Web Services and JPDL was developed to support Java applications. Because technological capabilities of the executing device cannot be determined in advance, mobile processes need an even higher abstraction level. Therefore, the most appropriate language to describe mobile processes turned out to be the meta-description-language XPDL. As this language is originally intended to be an abstract interchange format for different workflow engines and as it supports arbitrary platforms, it provides a very general view of processes. Unlike most of the other process description languages, it is also able to integrate manual tasks. Due to its resulting flexibility and extensibility it is therefore chosen as the basis for the enhanced approach as presented in section 4.

Another important requirement arises from the fact that mobile process execution always relies on contextual information. Consequently, context modeling and context data acquisition are crucial for the respective concepts and system infrastructure to be developed. Because mobile processes can involve different contexts which cannot be determined in advance, the context model has to be generic and application-independent. In order to enable the exchange of context information, the relevant data has to be provided in a standardized way. Furthermore, adequate techniques for context management and sharing, such as discovery, transformation, evaluation, and administration of context information, are required. To

also ensure reusability and extensibility, the context model and the context management system should be conceptually decoupled. Finally, the entire context system has to respect the constraints resulting from mobility, e. g. by notifying registered applications in case of specified changes, by allowing a deactivation of temporarily unused modules, or by providing custom filter mechanisms (Kunze et al. 2008, pp. 462–463).

Considering these requirements it can be derived that peer-to-peer and ad-hoc infrastructures of portable devices provide the most challenging environment for mobile applications. Consequently, systems which move at least important parts of the context computation into the supporting infrastructure – such as e. g. *Solar* (Chen and Kotz 2002, pp. 6–8) and *Nexus* (Dürr et al. 2004, pp. 15–18) – and relatively heavyweight ontological approaches – such as the *Service-oriented Context-Aware Middleware* (Gu et al. 2004, pp. 2656–2660) or the *Context Broker Architecture* (Chen et al. 2003, pp. 183–184) – are not suitable to fulfill the requirements for mobile context-based cooperation. Other existing context management systems (e. g. the *Hydrogen Framework* (Hofer et al. 2003, pp. 292–302) or the *Java Context Awareness Framework* (Bardram 2005, pp. 98–115)) have at least some deficiencies w. r. t. a clear separation between context model and management system which influences their flexibility and scalability. Furthermore, mechanisms to filter relevant context information or to realize extended overall context change notification often remain unsupported (cp. Kunze et al. 2008, pp. 463–465). On the other hand, the idea to ensemble the context of an entity by federating local context clippings of entities within a particular vicinity as proposed in the Nexus project is very interesting. Furthermore, the abstract and generic definition of context and its data as used in the *Context Toolkit* (Salber et al. 1999, pp. 434–441) turned out to be suitable for mostly a priori unknown demands of mobile processes. Therefore, both concepts will be partly considered in the conceptualization of context-based cooperation which is presented in the following section.

4 Realizing context-based cooperation with mobile processes

Especially the close cooperation between process management and context infrastructure to distribute and execute the mobile processes lead to the need of a specifically adjusted model and service architecture. To explicitly consider the presented requirements and observations, the resulting system infrastructure consists of a process execution environment, a context management system, and a distributed registry for service discovery.

4.1 Describing and executing mobile processes: the process service

In order to overcome the deficiencies of existing approaches for describing mobile processes, a new concept of a process service was developed. It realizes the integration of process management into the architecture for context-based cooperation and is comprised of two parts: Process description and process execution. The first part is realized by an enhanced definition language to describe the mobile process as well as the user's and application's non-functional demands. Based on the decision for the meta-description language XPDL (cp. section 3) the *DEMAC Process Description Language (DPDL)* has been developed to define a sequence of activities and constraints for their execution. DPDL extends and thus inherits the structure and those constructs of XPDL which turned out to be suitable for describing mobile processes (cp. **Fig. 5**). Most importantly, DPDL allows for a distributed handling of the process over heterogeneous systems. Thus, an entire process may be passed on to another device to continue working on the processes' tasks. Devices which are not capable of executing a particular task of the process can mark its latest execution state and search for other devices which can continue at the state reached so far. To make process descriptions as lean as possible and to achieve the required independence of technologies, DPDL allows specifying activities as *abstract service classes*. Within this concept, each specific functionality required to fulfill a specific task is determined by a short but significant identifier and the expected input and output parameters. A simplified example of such abstract representation is shown in **Fig. 4**. It contains the abstract representation of a short message service (SMS). The depicted *UUID* represents the *Universally Unique Identifier*, meaning this service will send a short text message. The *FormalParameters* define the mode and data type of the expected input and output parameters, i. e. the receiver's phone number and the message text as input parameters. The specified *Ids* are then used to refer to this functionality within the control flow, e. g. to avoid multiple specifications of the same service in case of the repeated use of an activity.

Depending on the capabilities of the mobile device that is currently responsible for the process, the abstract service identifier is resolved and mapped to accessible services. If, e. g., the mobile device supports the execution of Web Services, it may use suitable Web Service resources. If the mobile device uses other service invocation mechanisms, it is able to access other services providing the same functionality by using alternative protocols.

To narrow the selection of potentially participating devices and services according to the user's demands, the process description may also contain a set of non-functional parameters (*Strategies,* cp. **Fig. 5**). Strategies can be bound to individual activities or to the entire process. This way, the process service can ensure that only those services and devices are involved in the processes' execution that meet the specified requirements of the initiator. Examples are the specification of cost restrictions for the execution of the overall process or the limitation of potential process participants to a predefined group, e. g. employees within the same company.

The second part of the process service is an execution engine which resolves and executes mobile processes. It can either invoke activities locally or delegate the execution to a remote process service. When delegating a process, the description and all necessary data is transferred to the remote unit by using a basic transport infrastructure (cp. Kunze et al. 2006, p. 37). The execution of processes on mobile computing systems requires an efficient use of the available system memory. Therefore, the execution engine's architecture provides the ability to customize a compact kernel by plugging in functional modules to adapt to the capabilities of the underlying device (**Fig. 6**): A core migration module provides basic functionality – such as receiving, stor-

ing, and forwarding process descriptions.

A more powerful control flow execution module is responsible for executing the tasks of the process. It uses the migration component to communicate with other devices and can be enhanced by further device-specific extension modules.

4.2 Context modeling and management: the context service

The context service collects and maintains all information about the environment of a device. It filters and partitions the information, providing only those data needed to support process execution and migration. These are, among others, information about reachable devices and their services, location parameters, and data about users and their respective identities. For acquiring context information, a federated approach is chosen: Every device provides only local context information – and the information of all devices in the environment is merged in order to obtain the overall context.

To realize this, the developed context model (**Fig. 7**) provides standardized interfaces and formats to integrate different context resources: An *Entity* represents a single object which can aggregate several *Attributes* to describe contextual information on a higher level. This could be, e. g., a person with her location and activity – or a meeting combining a set of persons. On a higher level, the component *DomainContext* represents a self-contained and specifically demarcated environment which holds a limited number of relevant entities. This classification has the advantage to serve as a filter mechanism in a way that in a given context only that information is being selected which is relevant to the respective application. As a result, the number of context properties discovered and held by the mobile device is as small as possible and the amount of necessary information is customized to the particular environment of the application. The whole set of relevant *Domain Contexts* and information about the current domain is part of the *Device Context* which aggregates all context data of a single mobile system. The overall context contains the *Local Context* and the *Remote Context* of foreign (but locally reachable) mobile systems. As a result of the generic model, the remote context can contain arbitrary

```
Application Id="ShortMessageService"
    UUID="AA4F3DACB6C540AB8DF1A1D8BBC48894"
    FormalParameter Id="Receiver" Mode="IN" DataType="Phone-Number"
    FormalParameter Id="Message" Mode="IN" DataType="String
```

Fig. 4 Example of an abstract service description in DPDL

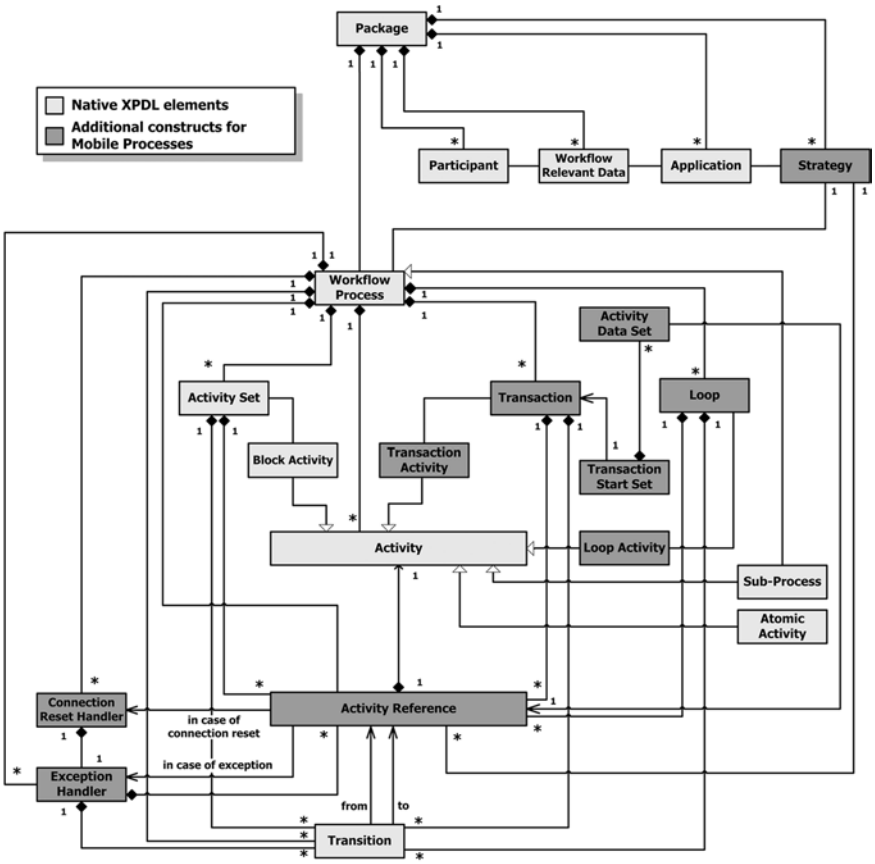

Fig. 5 DPDL language elements (Kunze et al. 2006, p. 39)

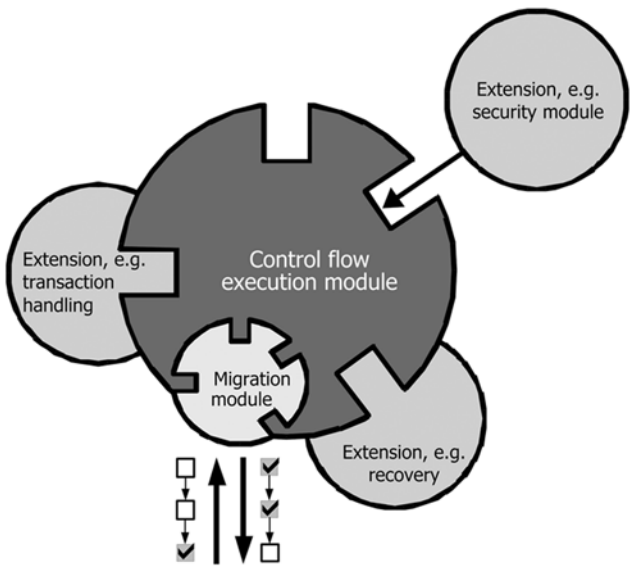

Fig. 6 Mobile process execution engine

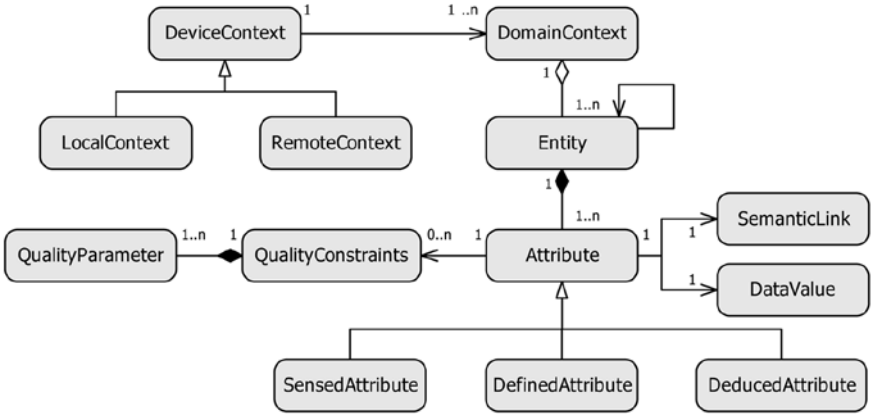

Fig. 7 Generic context model (Kunze et al. 2008, p. 467)

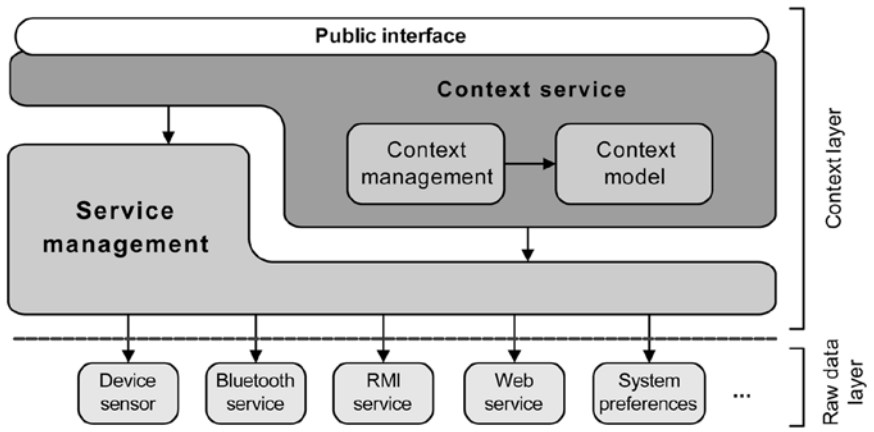

Fig. 8 Context management system (Kunze et al. 2008, p. 465)

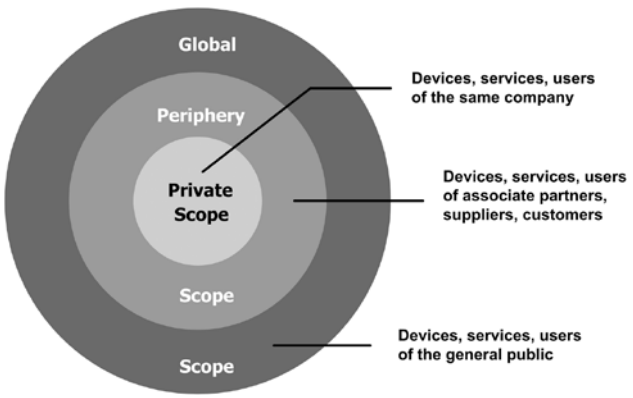

Fig. 9 Exemplary definition of a privacy scope

domains and entities, unaffected by the (limited) set of contextual information modeled within the local context.

Fig. 8 provides an overview of the integration of the presented context model into the coarse architecture of the context service. The *Raw Data Layer* provides interfaces to physical sensors of the mobile device itself or to logical sensors based on (remote) services. These components can be composed arbitrarily and can also change during runtime. Thus, the *Service Management* of the context layer is responsible for abstracting from these single resources in order to overcome their heterogeneity. To achieve the required separation between context model and management, all middleware functions concerning the *Context Management* are realized by autonomous services. Examples are the *RemoteContextService* which manages the exchange of context information between cooperating devices, the *TransformationService* which supports the conversion of contextual data values to equivalent formats, or the *NotificationService* which observes the local context model and the relevant context of remote systems in order to allow proactive actions through event notifications.

As the concept of context-based cooperation is heavily dependent on the willingness of devices and users to cooperate, a very important aspect is the consideration of privacy concerns. Therefore, the context model allows determining specifically which pieces of information may be revealed to other participants and which conditions must be fulfilled to share functionalities and resources. The specification of privacy requirements can be attached to the entities of the context model or to specific service instances. **Fig. 9** shows one of several possibilities to restrict resources by the definition of a corresponding privacy scope. The integrated *SecurityPrivacyService* then checks whether a requesting device has the proper identity to access such constrained data or functionality.

The presented context management services are also accessible by other components of the middleware architecture, e. g. by the process service which uses the context information in its execution strategies and decisions. Therefore, it is also possible to restrict the execution and migration of processes to specific participants, for example to employees within the same company or to paying customers only.

4.3 Service discovery and execution

As the type of services and applications provided in the environment cannot be determined at design time – but result from the available resources of the context during execution – the required activities within a mobile process are determined by abstract service classes to refer and identify applications and services in a technology-independent way (cp. section 4.1). To find and resolve these services, the context service contains a distributed registry which uses peer-to-peer mechanisms to obtain its knowledge. Therefore, each device holds a local service registry describing the functionalities provided by this node. Each time an activity has to be executed, a service query is initiated, containing information about the required functionality, about supported technologies and, optionally, a set of non-functional parameters from the process description. In case the local registry does not contain

an entry for a suitable service, the service query is passed to other devices. If at least one suitable service can be found, detailed information (such as their technology-specific description) is returned to the requestor. **Fig. 10** gives an overview of the corresponding architecture of the service management component together with the communication protocol to execute service queries remotely.

Once an adequate service has been found, its description is passed to the process service which is then responsible for the execution. Depending on the specific technology, the service description is interpreted and processed. For example, in case of invoking a *Web Service* or a *Java RMI service*, appropriate stub classes are generated and the (remote) service is finally called by invoking a respective adapter object with the arguments specified in the process description.

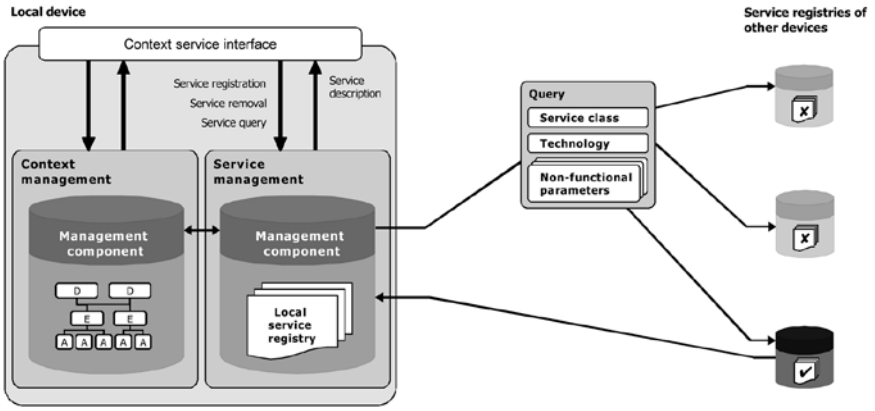

Fig. 10 Distributed registry: Answering a service query

5 Challenges and enhancements for mobile process business appliances

The approach developed so far provides a conceptual basis for specifying complex tasks in terms of mobile processes in order to overcome the limitations of isolated mobile systems by using shared resources and making use of the heterogeneity of the mobile environment. Mobile processes can be executed and are able to migrate according to non-functional requirements of the mobile user and to the respective available context. However – mostly due to the peer-to-peer aspects of the approach – there are still open problems with supporting business activities by mobile processes. This section identifies potential challenges and discusses respective management function enhancements for mobile process business appliances.

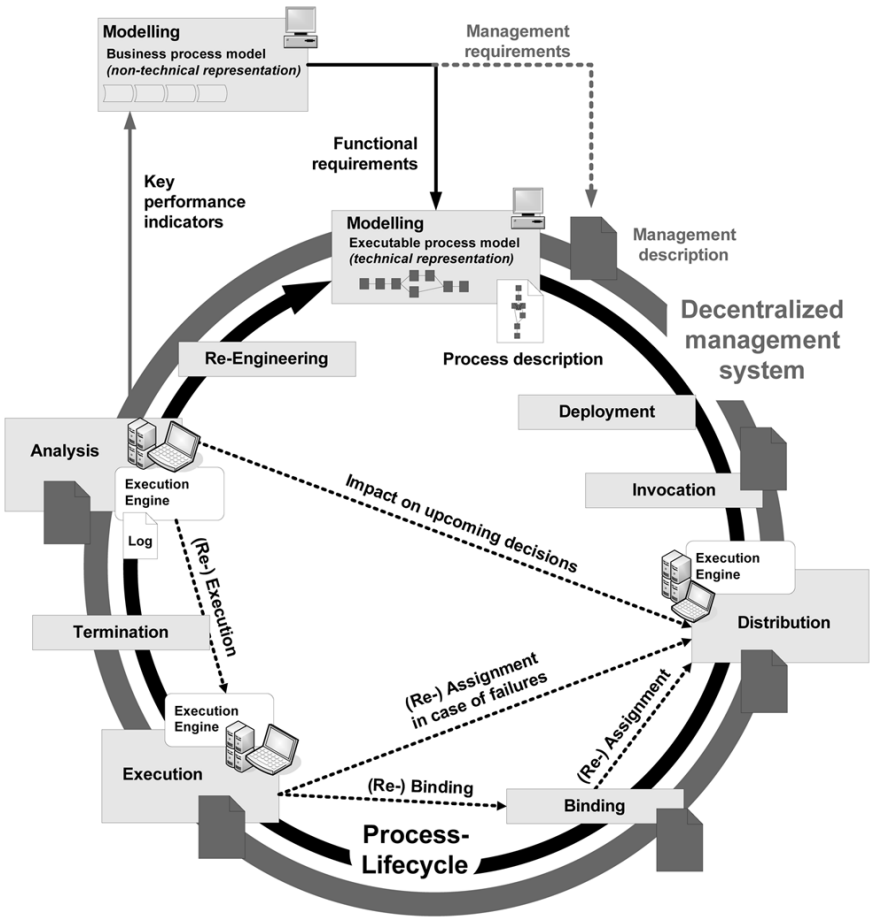

Fig. 11 Mobile process lifecycle management

Controllability and traceability
As mobile processes often leave the user's sphere of control, there is no way to get information about their execution state or to influence a running process instance retroactively, e. g. for cancelling it. In addition, there is a need for transparency and traceability, e. g. logging for accounting purposes.

Recovery mechanisms
Especially if an executing device collapses permanently, there is a residual risk of losing the mobile process somewhere in the mobile vicinity. The provision of recovery mechanisms, such as generating controlled copies of the mobile process, increases the probability of a reliable process execution.

Transaction processing and compensation
In order to guarantee reliable and complete process execution, parts of a process should not be processed while associated other process activities are not. Therefore, in case of failures, it may be necessary to compensate such activities which have already been carried out or to retry executing the failed task by other participants.

Security
Business activities often contain confidential data and must therefore be

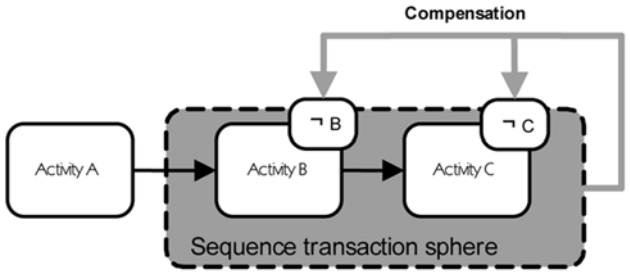

Fig. 12 Sequence transaction

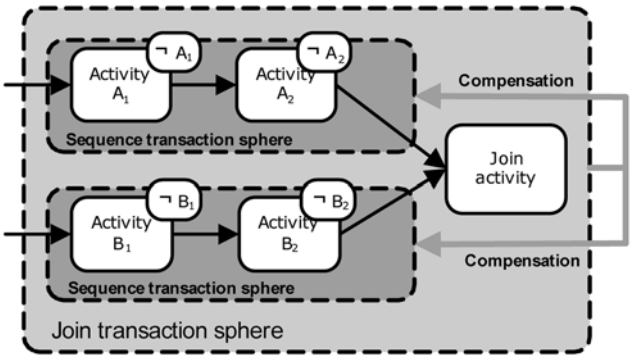

Fig. 13 Join transaction

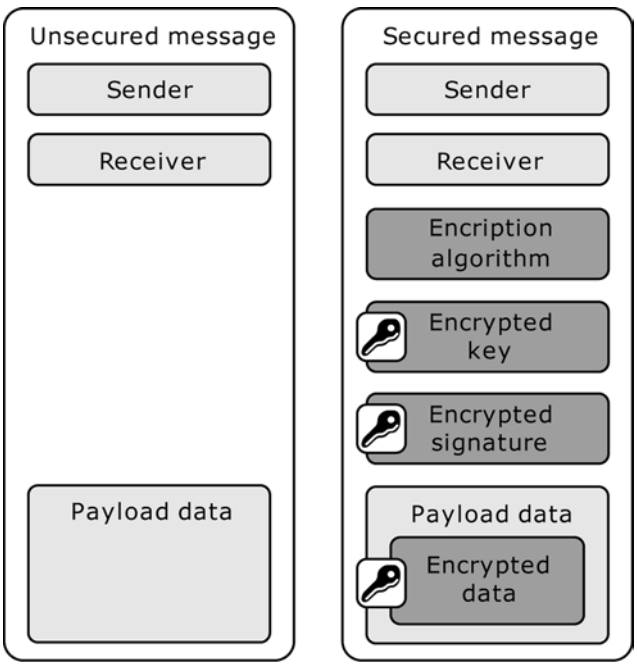

Fig. 14 Structure of unsecured and secured message exchange

protected against unauthorized access and manipulation. If the mobile process migrates to other network participants, their respective identities may have to be verified and data transfer must allow for encryption and integrity checking.

Performance

Due to the fact that migration concentrates on the processes' next activity only, the path actually taken by the mobile process might be suboptimal. Furthermore, the workload of single participants remains unconsidered, so, e. g., an important mobile process may have to wait until prior tasks have been finished.

5.1 Advanced process management

In order to fulfill controllability, traceability, recovery, and performance requirements, an advanced lifecycle management of distributed processes has been elaborated. While lifecycles of traditional (business) processes are characterized by rather isolated phases, the dynamism of the mobile environment and the distribution of resources require interconnecting the lifecycle phases of such processes for data acquisition and flexible reactions at any time. Furthermore, as devices already cooperate in order to execute a common task by migrating processes, they should also cooperate for allowing a decentralized management of these processes.

Based on these observations, the resulting lifecycle for managing mobile processes is depicted in **Fig. 11**. At design-time (i. e. in the modeling phase), general management rules which should be enforced by all participating devices are specified and attached to the process description. When the process is deployed and invoked, these rules become active and influence the way the (mobile) process is handled by its executors.

The first phase actually contributing to this decentralized management is the phase of *distribution*. Within this step, the upcoming task is assigned to a (mobile) participant responsible for its execution. Depending on the devices' capabilities, the dynamism of the network and the complexity of the control flow, also process participants for follow-up activities can be predetermined. To realize this, the process description is analyzed and requirements of upcoming activities are compared to the current context data. The result is a proposal for a temporarily optimal execution strategy, involving migration to other devices based on their local and remote context. Thus, dead-end situations and unnecessary migrations can be avoided (e. g. executing the next three activities at the same device instead of migrating the process three times to execute the activities individually).

The next step contains the *binding* of the selected participant. This can either be the assignment of a specific activity or the migration and execution of the entire process. To increase performance, it is also possible to reserve execution time prior to the actual invocation, so the required resources can be allocated in advance. For example, the devices participating along the calculated migration path can be bound at the time of their selection and invoked later. Nevertheless, if the environment changes and the determined path becomes invalid, the reservations have to be cancelled and the migration path has to be recalculated – which means the lifecycle has to be rolled back to the phase of distribution for the remaining activities.

The *execution* phase takes care of the proper processing of activities. There-

fore, the decentralized management system is responsible for monitoring, logging and recovery of mobile processes. While the original process remains untouched, a partial or entire copy containing the required status information is sent to the cooperating managing network, allowing customized access to this process instance. Furthermore, logging and result data can be collected and returned to a specified participant directly or by migration. If the executing device itself collapses permanently and therefore fails to execute the process before a specified deadline, a backup device can carry on using the copy from its last checkpoint. Thus, execution is again interconnected with the distribution phase in order to make a new assignment where required.

The last phase of the process lifecycle contains an optionally post-mortem *analysis* of single activities or of the entire process, e. g. to integrate a target-performance comparison or to gather information about process participants if necessary. The information gathered by such analysis can serve as valuable input for future assignment decisions. Furthermore, the results of this analysis can take effect on the process template, e. g. to improve it for future applications.

5.2 Transaction processing

As a mobile process may span several devices and the execution of an upcoming activity may depend on these prerequisites, it is – in most cases – not possible to temporarily lock the effects of a prior activity in order to ensure transactional behaviour following the *ACID* concept (cp. Elmasri and Navathe 2004). Therefore, transaction handling for mobile processes concentrates on so-called *long running transactions* and in particular on the workflow-specific concept of compensation spheres as introduced by Leymann and Roller (2000, pp. 259–274). A traditional compensation sphere is a collection of activities specifying pieces of work which are tightly related. If any of these activities has not been performed correctly, all other activities that have already been performed must be compensated by a set of counterpart activities.

The compensation sphere concept for mobile processes consists of Sequence- and Join Transactions. Sequence Transactions (**Fig. 12**) can be applied to control flow structures without parallel execu-

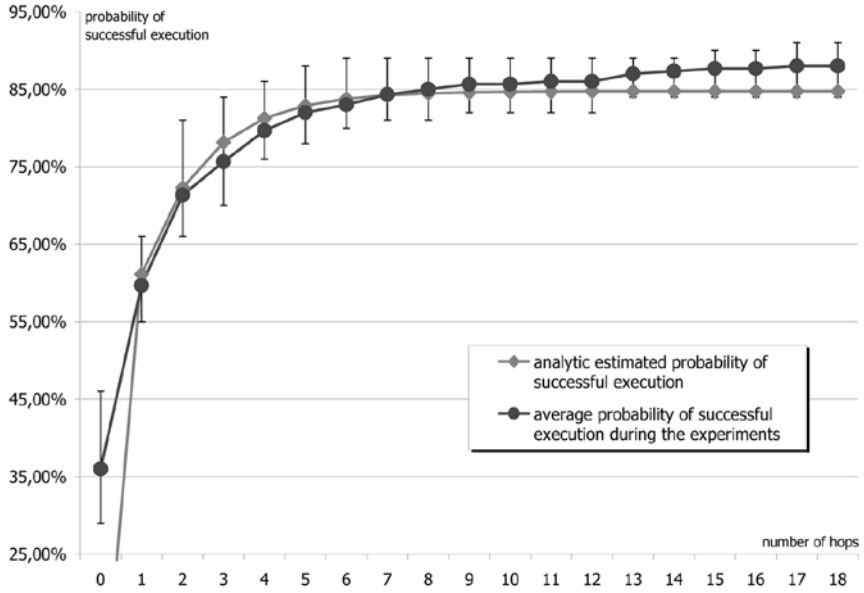

Fig. 15 Results of the experimental evaluation (Kunze et al. 2008, p. 469)

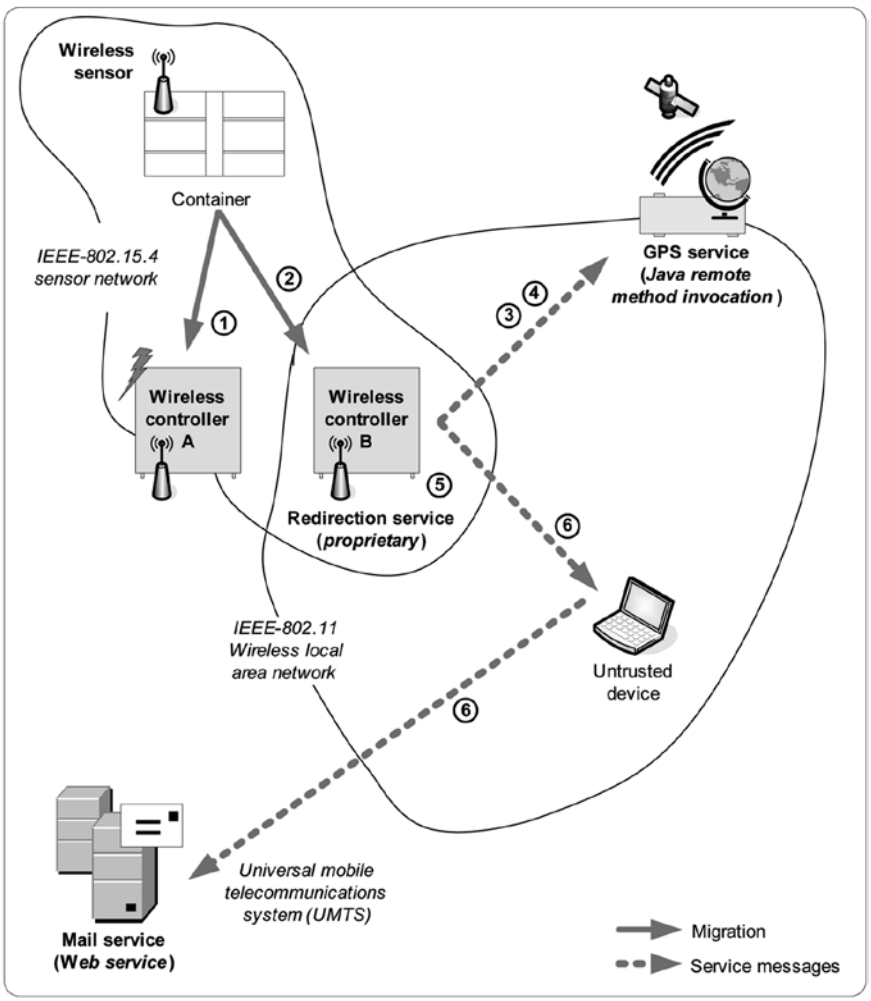

Fig. 16 Transport logistic network

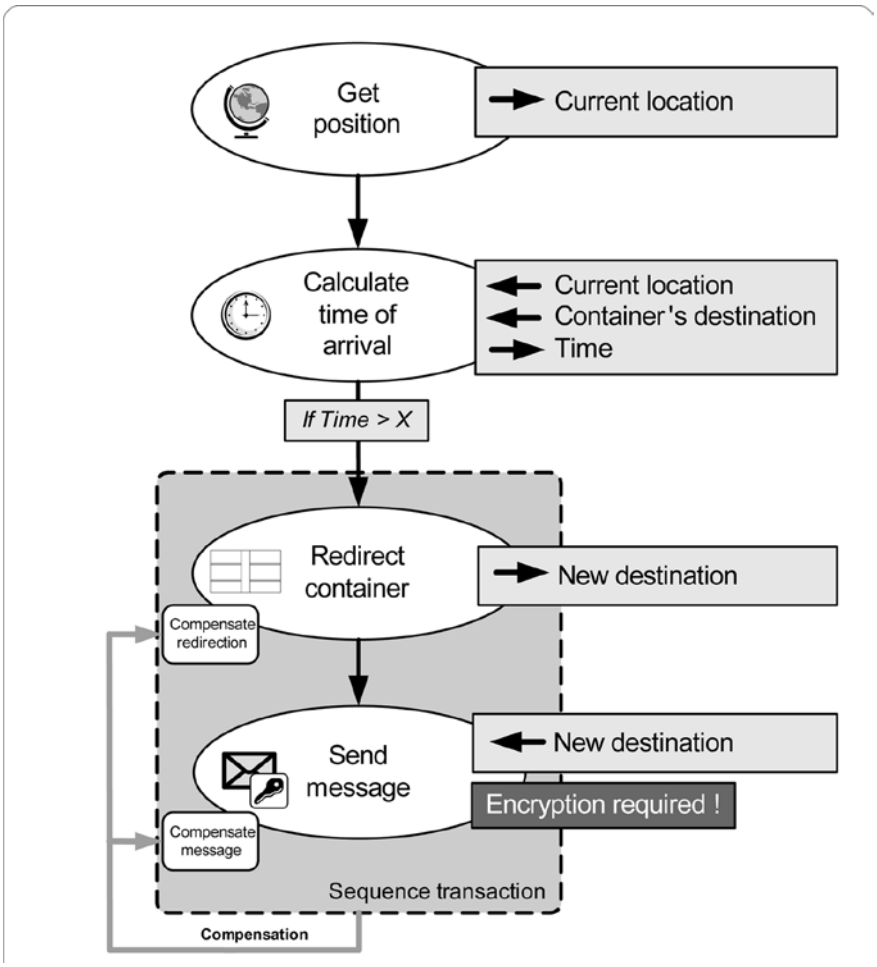

Fig. 17 Mobile process example

tion. To allow adequate backward recovery mechanisms, all initial and intermediary data accumulated during execution has to be recorded in a log file. This involves data about migration partners as well as executed services in order to enable a proper compensation in case of failure. On the other hand, Join Transactions support parallel execution paths (**Fig. 13**), making use of the fact that parallel paths of mobile processes can be joined at a determined synchronization point. Therefore, the results of single execution paths finally reach the synchronization point and allow for a decision about the necessity of compensating one or more paths. As Sequence and Join Transactions can be mixed, even complex transactional structures can be represented by such compensation spheres. Furthermore, the propagation of interim results also allows forward recovery, avoiding loss of work already (partially) performed.

5.3 Security

To realize secure communication between two collaborating systems, the messaging layer of the middleware includes an (optional) security component which provides mechanisms for so called *end-to-end security* (as e. g. proposed in the *Web Service Security specification* by Nadalin et al. 2006). End-to-end message level security is established when a message that traverses multiple stationary and mobile devices is secure over its full route – although intermediaries without security support may potentially participate in the exchange of secure messages. The structure of such a message is depicted in **Fig. 14**. The unsecured transport message contains the mandatory elements for the sender, the receiver and the payload container. It is extended to include the type of encryption algorithm used to encrypt the payload data, the key information for symmetric decryption and a digital signature of the sender. As the resulting secured transport message provides the original routing elements in an unencrypted way, the message can still be delivered.

The encryption procedure itself is hybrid, i. e. it uses symmetrical as well as asymmetrical algorithms. Because asymmetrical encryption methods are more complex than symmetrical ones, only the key required for decrypting the symmetrical encoded payload is encrypted with the more complex *public key* method. This relieves less powerful mobile devices from complex computations needed to encrypt and decrypt large payload data. However, the specific type of algorithm can be chosen due to individual security policies, providing extensibility also for future algorithms.

To protect the distributed execution of a mobile process, the initiating user can thus define a set of non-functional criteria, specifying, for example, the kind of encryption algorithm, and attach it to the process. End-to-end message security therefore allows the mobile process to be migrated until an authorized participant is found and ensures integrity and confidentiality of messages even through unauthorized intermediaries.

6 Experimental and scenario-based evaluation

In order to evaluate the applicability of the generic context model and management system as proposed above, a prototype implementation of the presented components has been realized (cp. Kunze et al. 2008, pp. 467–469). In order to verify the assumptions introduced in section 2, the evaluation includes an experiment to determine the probability of successful execution. The environment for the experimental series consists of a simple process with one single activity, six heterogeneous devices with two devices having the capability to execute the processes' activity, and four devices unable to do so. Because sender and receiver of the mobile process cannot be the same, there are 5 possibilities for each process to migrate from one device to another. This leads to an execution probability of $p = 40$ % within the entire system. To test the behavior of the prototype under load, several test runs were carried out, each including 100 processes. **Fig. 15** shows the average number of hops resulting from migrations necessary to execute the process successfully compared to the expected analytical value. The analysis of the experiments further shows

that only a few hops suffice to increase the probability of successful execution to levels more than twice as high. The estimated probability and the applicability of the presented concept can therefore also be confirmed by practical experimentation.

To test the applicability of the developed concepts, also a complementary, scenario-based evaluation was carried out covering selected optional components of the mobile process management system (i. e. transactions, security and mobile process management). Experiences and observations reported are based on applying the prototype to a use case from the area of transport logistics as introduced in section 1. **Fig. 16** shows the exemplary network infrastructure of a trans-shipment centre for container traffic, where a freezer container is monitored by a wireless sensor. In case of a malfunction of the cooling system, the wireless sensor instantiates a predefined mobile process template which specifies reactions to the detected situation. The resulting process instance is depicted in **Fig. 17**, showing a selected set of abstract activities and their input/output data: First, the current position of the container has to be acquired (*Get Position*). Second, the estimated time of arrival has to be calculated in order to decide whether the cargo will thaw until the container arrives (*Calculate Time of Arrival*). If the time until arrival is considered too long (*Time >X*), the container must be redirected, e. g. to a cold storage (*Redirect Container*). Furthermore, a message has to be generated to inform maintenance support where to find the defect container (*Send Message*). The last two activities of the process are realized as a transaction because the engineer will probably not be able to find the container without knowledge about its new destination. Therefore, the redirection has to be undone if the message cannot be sent within a specified deadline. Furthermore, the message's information must be transferred encrypted – which is attached as a non-functional requirement.

Due to performance restrictions, the wireless sensor is not able to execute the process itself. As the process will therefore leave the sensor's sphere of control, it attaches a management descriptor which holds rules about its recovery, monitoring and logging requirements. In this use case, the management descriptor specifies that process execution should be monitored by a backup-device and that, in any case of irregularity, the process should be restarted by this device. Furthermore, process participants, failing devices and recovery actions should be logged and failing devices should be excluded from further process execution. If applicable, a context-based look-ahead procedure should be used to find the most appropriate migration path in order to avoid unnecessary migrations.

A possible execution path of the mobile process is shown by the numbered arrows in **Fig. 16**. The wireless sensor is not able to calculate a temporarily optimal execution strategy for the process. Therefore, it migrates the process to an arbitrary other device in its communication range, in this case to wireless *Controller A* (step 1). But *Controller A* has a malfunction and is not able to call any other service to execute the process. A timeout indicates its failure and the process is restarted. As this incident is also logged, the failing device is avoided during upcoming migrations and, consequently, in the second attempt *Controller B* is selected (step 2). As this controller is a stationary and quite powerful device, it is able to call a nearby GPS service to collect data about its current position as well as to calculate the estimated time to arrive at the container's destination (steps 3 and 4). Furthermore, it can decide about the necessity of redirection and uses its own local service to unload the container. However, as it is not connected to the Internet, it has to use an intermediary device to call an appropriate e-mail service. The message is therefore encrypted as described in section 5.3. Furthermore, as the use of the network (e. g. UMTS) causes telephone charges, participant and payment details can be logged to the mobile process and can be refunded later.

7 Conclusion and future research

Although prices for computer hardware are constantly decreasing, the cost-value ratio of many mobile devices for executing highly individual applications is still insufficient for many advanced business applications. Therefore, the concept of mobile processes has been introduced which helps to combine both mobile and stationary resources whenever possible. Context-based cooperation enables structured ad-hoc applications by integrating arbitrary services, devices, and users in mobile processes taking advantage of their combined capabilities. The sup-

Abstract

Sonja Zaplata, Christian P. Kunze, Winfried Lamersdorf

Context-based Cooperation in Mobile Business Environments – Managing the Distributed Execution of Mobile Processes

Realistic requirements of mobile business applications often exceed the capabilities of their respective local environments. In order to overcome such restrictions of specific mobile devices, services, and resources, this contribution introduces the concept of context-based cooperation. It is based on mobile processes which enable applications to cross boundaries of individual systems and thereby allow combining both mobile and stationary resources in order to realize highly dynamic individual applications. This contribution presents an approach for realizing context-based cooperation built upon on a respective context management infrastructure and execution environment. It also identifies specific requirements and proposes related enhancements for mobile business applications.

Keywords: Business process management, Mobile computing, Context-awareness, Distributed systems, Service-oriented computing, Mobile process, Cooperation, Migration

porting middleware infrastructure is based on a context management system and a process execution environment to define a sequence of tasks which can be executed in a distributed way by (possibly) migrating processes to other mobile or stationary devices. Such a strategy makes accessible resources in the mobile environment available for individual applications and, therefore, increases the execution probability of tasks which could not be executed on a single mobile system with static resources only.

For applications of mobile processes to more sensitive business use cases, specific reliability and controllability requirements have been identified and, accordingly, important enhancements for the mobile process management as well as for transaction handling and security mechanisms have been proposed. Therefore, within the boundaries of a certain business domain (e. g. within a company or an affiliated group), mobile processes provide reliable and secure ways to enhance mobile cooperative applications. Nevertheless, in current open system infrastructures, a component's willingness for additional trust and cooperation efforts might still be low without an additional (e. g. economic) stimulus. Therefore, future work in this area has to include, e. g., an accounting mechanism as a basis for compensating third party users for providing resources or for participating in the execution of foreign processes, making also global cooperation scenarios crossing multiple business domains more profitable.

References

Adelstein F, Gupta SKS, Richard III GG, Schwiebert L (2005) Fundamentals of mobile and pervasive computing. McGraw-Hill, New York

Arkin A, Askary S, Fordin S, Jekeli W, Kawaguchi K, Orchard D, Pogliani S, Riemer K, Struble S, Takacsi-Nagy P, Trickovic I, Zimek S (2002) Web Service Choreography Interface (WSCI) 1.0. Specification, World Wide Web Consortium

Bardram JE (2005) The Java Context Awareness Framework (JCAF) – A service infrastructure and programming framework for context-aware applications. In: Gellersen HW, Want R, Schmidt A (eds) Pervasive computing, pp 98–115. Springer, Heidelberg

Barreto C et al. (2007) Web Services Business Process Execution Language version 2.0, Specification, OASIS

Chen H, Finin T, Joshi A (2003) An intelligent broker for context-aware systems. In: Adjunct proceedings of Ubicomp 2003, Seattle

Chen G, Kotz D (2002) Solar: A pervasive computing infrastructure for context-aware mobile applications. Technical report, Department of Computer Science, Dartmouth College Hanover

Dürr F, Höhnle N, Nicklas D, Becker C, Rothermel K (2004) Nexus – A platform for context-aware applications. In: Roth J (ed) 1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuvS, Hagen

Elmasri R, Navathe SB (2004) Fundamentals of database systems, 4th edn. Addison-Wesley, Boston

Fleisch E, Dierkes M (2003) Ubiquitous Computing aus betriebswirtschaftlicher Sicht. WIRTSCHAFTSINFORMATIK 45(6):661–620

Gu T, Pung HK, Zhang DQ (2004) A middleware for building context-aware mobile services. In: Proceedings of IEEE vehicular technology conference (VTC), Los Angeles

Hofer T, Schwinger W, Pichler M, Leonhartsberger G, Altmann J, Retschitzegger W (2003) Context-awareness on mobile devices – the Hydrogen approach. In: Proceedings of the 36th annual Hawaii international conference on system sciences (HICSS'03), Big Island

JBoss (2005) JBoss jBPM 3.0 – Workflow and BPM made practical. Documentation, JBoss Company

Kunze CP, Zaplata S, Lamersdorf W (2006) Mobile process description and execution. In: Eliassen F, Montresor A (eds) Proceedings of the 6th international conference on distributed applications and interoperable systems, pp 32–47. Springer, Heidelberg

Kunze CP, Zaplata S, Turjalei M, Lamersdorf W (2008) Enabling context-based cooperation: A generic context model and management system. In: Abramowicz W, Fensel D (eds) 11th International conference on business information systems, pp 459–470. Springer, Heidelberg

Leymann F, Roller D (2000) Production workflow – concepts and techniques. Prentice Hall, Upper Saddle River

Nadalin A, Kaler C, Monzillo R, Hallam-Baker P (2006) Web services security specification. Specification, OASIS

Papazoglou MP (2003) Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the fourth international conference on web information systems engineering (WISE 2003), Rome

Riemer K (2001) Business process specification schema. Specification, OASIS

Satyanarayanan M (1996) Fundamental challenges in mobile computing. In: Proceedings of the 15th ACM symposium on principles of distributed computing, Philadelphia

Salber D, Dey AK, Abowd GD (1999) The context toolkit: Aiding the development of context-enabled applications. In: Conference on human factors in computing systems (CHI 99), Pittsburgh

WfMC (2005) XML Process Definition Language, version 2.00. Specification, Workflow Management Coalition