Towards Mobile Process as a Service

Sonja Zaplata
Distributed Systems and Information Systems
University of Hamburg
Vogt-Kölln-Strasse 30
22527 Hamburg, Germany
zaplata@informatik.uni-hamburg.de

Winfried Lamersdorf
Distributed Systems and Information Systems
University of Hamburg
Vogt-Kölln-Strasse 30
22527 Hamburg, Germany
lamersdorf@informatik.uni-hamburg.de

ABSTRACT

Process as a Service (PaaS) addresses modeling, execution and management of business processes without running extensive and costly process management software. Such a flexible outsourcing strategy is especially advantageous in the context of mobile devices and services which are increasingly relevant for contemporary business activities. Based on the concept of *context-based cooperation*, this paper proposes a PaaS solution for mobile participants which enables them to share existing local and remote resources and to utilize PaaS functionality of cooperating providers in a user-defined way. The approach is realized and evaluated by an extended prototype implementation of the *DEMAC (Distributed Environment for Mobility Aware Computing)* platform.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

Keywords

Mobile Business Process Management, Cloud Computing, Process as a Service, Mobile Computing

1. INTRODUCTION

The acquisition and operation of specialized process management software including a process engine to automatically execute business processes typically implies high expenses and efforts. Associated costs (e.g. for software licenses, maintenance, support and training of employees) only pay off if many complex processes have to be managed in short time. However, for the execution of few, non-recurrent, or ad-hoc processes the purchase of a complete business process management solution is often uneconomical. As an alternative, process automation can be realized on the basis of the *Software as a Service (SaaS)* paradigm [14]. In the case of process management, the software (i.e. the process engine) is operated by an external service provider and utilized by several customers in order to reduce capital investments and running costs. Therefore, process initiators only hold few domain-specific process templates which can be adapted to special purposes and are then passed to the service provider to be executed. As this strategy can be used to outsource several aspects of process management, the provision of such functionalities is often referred to as *Business Process Management as a Service* or short *Process as a Service (PaaS)* [4].

In order to integrate field staff or valuable context-aware mobile applications, also the need for and the emergence of *mobile process management systems* increases (e.g. [6, 11, 13, 5]). For example, mobile users or applications often require to initiate (ad hoc) business processes from their current location, e.g. by taking an urgent order at the customer's site, or to access specific mobile components such as sensors, RFID readers, GPS devices or cameras which often cannot be integrated by traditional stationary process management systems. However, similar to the aspects discussed above, the individual acquisition and operation of process management systems for mobile devices are often unfavorable - especially as the initiation and execution of processes is often unfrequent and mobile process management systems are usually not operated at full capacity. Thus, the usage of PaaS or sharing of existing (private) process engines is again an attractive alternative.

The contribution of this paper is thus to propose a distributed PaaS approach which fits the decentralized nature of mobile systems by integrating the capabilities of (private) mobile resource providers and allowing them to form a dynamic alliance for each upcoming business process. However, the delegation of (mobile) process execution to a distributed mobile PaaS system has advanced requirements regarding traceability, controllability, process privacy and accounting. The approach presented here is based on the existing concept of *context-based cooperation* which is introduced in the following section. Section 3 presents the developed enhancements in order to use context-based cooperation as a basis to build a distributed mobile PaaS system. Section 4 presents experiences gained with the prototypical implementation and Section 5 distinguishes the achieved results from related work. The paper concludes with a short summary and outlook.

2. BACKGROUND

Most small and medium-sized mobile devices such as sensors or cell phones are not capable to host a complex business process management system, because they are restricted in

SAC'10 March 22-26, 2010, Sierre, Switzerland.

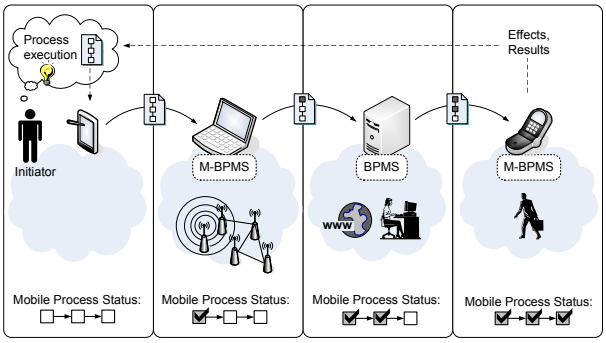

Figure 1: Context-based cooperation (cp. [10])

computing power and memory, are often temporarily without network connection and cannot access complex server applications due to technological constraints. Furthermore, mobile applications and devices are often designed for a special purpose and are thus very heterogeneous [12], and mobile resources are often decentralized and can only be accessed by using local networks (e.g. a sensor network). As a consequence, also more powerful mobile devices often do not have the capability of executing an entire business process, but are potentially able to contribute with the execution of a particular subset (i.e. an activity or a subprocess) [10].

The concept of *context-based cooperation* represents an approach to support the execution of such complex application tasks by using context knowledge in order to integrate the capabilities of different mobile nodes of the vicinity. The following subsections introduce the existing concepts, show a respective use case scenario and identify further requirements.

2.1 Context-based Cooperation

The concept of context-based cooperation focuses on the delegation of so-called *mobile processes* to enable the cooperative execution of a given structured task in a mobile environment. A mobile process is defined as a piece of mobile code representing a goal-oriented composition of services which can be migrated to other (mobile or stationary) devices in order to share the functionality provided by these nodes [10]. Figure 1 shows a process initiator who is himself not capable of executing an ad-hoc process requiring several mobile and stationary services. Depending on the process's tasks and its technological requirements it has to involve other process engines in order to execute the process's activities, e.g. to access local functionalities of a sensor network and mobile users as well as stationary web services and desktop users. The opportunistic strategy involves three stages: As long as an activity can be executed *locally* by an application running on the same (mobile) device, the device is responsible for its execution. If such local applications are unavailable, the device can search for adequate services provided by other devices in its vicinity to execute the task *remotely*. In case the direct vicinity does not provide the required service, the process description can be *migrated* to another remote device in order to enable the execution in a different vicinity. It was shown that this strategy of process migration increases the probability that an (ad-hoc) process can be executed successfully [10] – especially in mobile environments where resources are often locally unavailable.

2.2 Use Case Scenario

As an example, Figure 2 shows a BPMN diagram of a business case extracted from a trading company. Purchasing agents of the trading company are often visiting fairs and exhibitions. As local exhibitors sometimes offer special prices, it is favorable to buy products directly on the spot. However, to ensure a good deal, it is necessary to compare the current market price of each item (e.g. by checking prices of competitors). If the purchase of an item is considered to be economical, it is added to the list of proposed items which must be confirmed by the chief purchase officer. Depending on his agreement, the local purchasing agent can make the deal.

Especially if the number of selected products is high, a semi-automatic and computer-supported execution of the depicted steps is advantageous. Using an executable representation of this business process (e.g. in WS-BPEL or XPDL format), the mobile purchasing agent involves suitable mobile devices (e.g. a barcode scanner or RFID reader) in order to collect the required data (e.g. product identification codes). Market prices are looked up and compared using web services available over the Internet. To be reachable everywhere and anytime, the chief purchase officer uses two proprietary applications running on his notebook in order to view proposed items and confirm their purchase.

However, installing and maintaining an individual specialized mobile process management system for each purchasing agent is very unprofitable. Furthermore, due to the (local) distribution and heterogeneity of involved services and applications, the usage of one single centralized process management system is also often not applicable (e.g. the barcode scanner is not connected to the Internet). The unnecessary repeated transfer of process data (e.g. as required for the second, third and fourth activity) using mobile communication networks should also be avoided. In this case, the migration of the entire process is also more efficient as the repeated invocation of a remote service. Thus, the flexible utilization and combination of available mobile and stationary process management systems and respective local services is advantageous for the execution of such business tasks.

The ad-hoc initiation of such processes in a mobile environment however requires a critical mass of service providers which are willing to share their resources (i.e. process engines and applications) and allow the execution of external processes. In the use case, several local exhibitors are running simple mobile process execution engines in order to accept and initiate orders which are placed by the visitors of

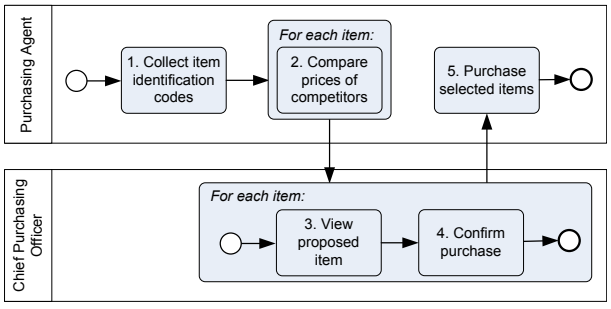

Figure 2: Business process of the example scenario

the fair. If a process engine is not utilized to full capacity, its resources can be shared - which has the advantage to reduce capital investment and running costs such as maintenance by an additional source of revenue, e.g. on a pay-per-use basis.

Due to the weak reliability assumed for distributed mobile systems, also the acceptance of the process initiator to submit his process to a (mostly transparent and unsecure) mobile cloud environment has to be strengthened. The next subsection summarizes this observations and identifies the most important requirements regarding the provider as well as the consumer side of such cooperations.

2.3 Requirements

By encapsulating the existing middleware platform for the execution of mobile processes and exposing its functionality of cooperative process execution "as a service", the concept of context-based cooperation forms a potential basis for a distributed mobile PaaS system. Nevertheless, sharing existing mobile resources has to be advantageous for both PaaS consumers and private operators of mobile process engines and gain sufficient acceptance from both sides. Furthermore, inherent characteristics of mobility have to be considered, leading to the following requirements:

- R_1 : The required solution must be able to ad hoc share and integrate resources which are potentially (temporarily) disconnected.
- R_2 : There must be a higher level architecture to attract and recruit a sufficient number of external actors, e.g. by providing incentives, and to include them in an overall payment and accounting model.
- R_3 : As the control flow of a distributed process leaves the client's direct sphere of influence, the process must be protected against unauthorized access.
- R_4 : Due to unreliability of mobile environments, there is an increased need for traceability and controllability of the process execution.
- R_5 : The individual interests of the mobile PaaS consumer must be respected - even if he is temporarily not available. Thus the process initiator has to make a detailed description on *how* the process has to be executed, i.e. to proactively influence where the process is executed.
- R_6 : Due to mobility and decentralization, central navigation nodes and runtime administration authorities should be avoided.
- R_7 : Limited resources of mobile devices require customized strategies for service invocations at runtime using context knowledge and preferably information about non-functional aspects such as service availability and load.
- R_8 : New services and service providers may enter or leave the system at any time and devices which will actually contribute in the execution of a process may not be known in advance [9]. The PaaS functionality should thus be discovered and picked dynamically, e.g. depending on the current context.

The following section makes a proposal for a further development of context-based cooperation for the application in mobile PaaS environments with a special focus on the aforementioned aspects.

3. MOBILE PROCESS AS A SERVICE

A distributed *Mobile PaaS (MPaaS)* system should allow to integrate (commercial or private) mobile and stationary resources offering the functionality to execute a subpart of a given process at their respective site. The concept presented here uses peer-to-peer process execution based on runtime process migration as presented in Section 2.1. An *MPaaS provider* is thus defined as a mobile or stationary device (potentially belonging to an external party) which runs process management software that can be utilized by *MPaaS consumers* in order to execute their business processes.

The resulting distributed MPaaS system architecture consists of a (traditional) public PaaS provider responsible for administration and accounting, a number of autonomous private providers which are able to execute a process in a cooperative and decentralized way, and a number of potential consumers which are interested to consume MPaaS functionality. Note that these roles can change flexibly, e.g. an MPaaS provider can also act as a consumer in case it uses MPaaS functionality of other systems to execute his own processes. Furthermore, the public PaaS provider can also act as a (stationary) MPaaS provider by executing processes with the "normal" PaaS functionality.

Figure 3 gives an overview of the overall concept. It is comprised of three phases: Phase 1 can be understood as a preparation phase in order to organize administration and to create an account for each participant. Besides consumers, also private providers which are interested in sharing their process engines and respective resources can register at the public PaaS provider (phase 1a) to be refunded later. This does not necessarily need a stable network connection, but can alternatively be realized offline, e.g. by a one-time human interaction. Also at this time, public keys and identities of participating parties can be exchanged if necessary (cp. Section 3.3).

Depending on the capabilities and the performance of the participating devices, three kinds of MPaaS participants can be distinguished:

- *Full participants* provide business process management functionality, i.e. participate in the execution of the process. This role is, in general, applicable for more powerful mobile devices and stationary systems.
- *Back-up participants* are used to monitor the execution of a process executed by a full participant. Therefore, back-up participants need basic management capabilities, such as receiving heartbeats, checkpoints and events sent by the monitored participant.
- *Proxy participants* act as simple carriers of a process description, i.e handing-over the process to a full participant or to a public PaaS provider. As this functionality is very simple, also small and less-powerful devices can take on the role as such a proxy peer.

In order to allow offline payment even in decentralized mobile ad-hoc-networks, the potential MPaaS consumer can

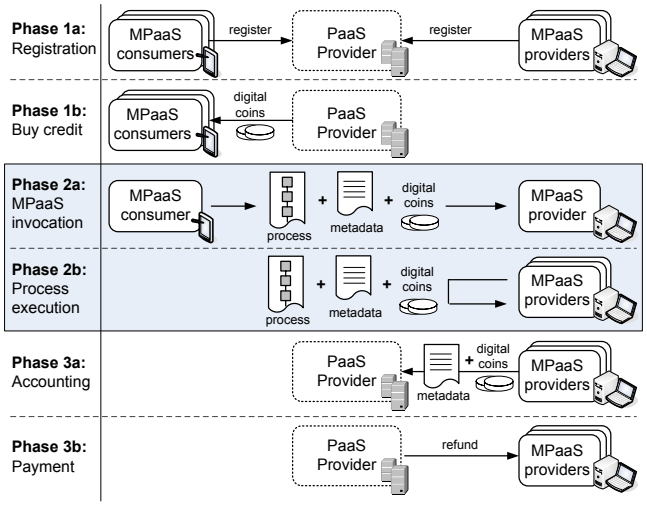

Figure 3: Overview of the MPaaS concept

buy a suitable number of *digital coins* (phase 1b) representing a pre-paid value and holding a signature of the issuing PaaS provider.

Phase 2 covers the runtime, allowing MPaaS customers to use the resources which are shared by the peer providers by initiating mobile processes. The process description together with a user-defined metadata document and a digital coin are passed to the selected MPaaS provider. By receiving the digital coin signed by the PaaS provider, the MPaaS provider can be sure that his contribution will be refunded later. If the resources of the provider are not sufficient to successfully execute the process, the process description and the digital coins are passed on to another MPaaS provider. As digital coins are forgery-safe, they cannot be duplicated or used in another context. Finally, all participating parties are recorded in the migration document's log file until the execution of the process is finished.

The last phase (phase 3) handles the delivery of individual log files, migration metadata logs and digital coins to the PaaS provider. If the process's activities have been successfully executed, the participating parties are rewarded by adding a certain amount of money to their account - while the shares depend on their actual contribution (i.e. as simple proxy providers or as full providers considering the number of activities which have been executed). Thus, individual log files of MPaaS providers must match the entries in the metadata document. In case process execution has (partly) failed, the customer's account can be issued a partial credit.

The remainder of this paper focuses on the second phase of the concept (cp. the highlighted part of Figure 3) which covers the processes' runtime. Therefore, the following subsections specify which additional information has to be considered in order to ensure both a distributed execution as intended by the MPaaS consumer as well as the necessary refunding of MPaaS providers.

An overview of the required metadata is summarized in Figure 4. In order to avoid influences or changes on the original process model as originally motivated by the underlying business process, the information necessary for migration is kept external to the process description and is only referencing its relevant parts, i.e. its activities. Technically this can be realized by an additional document holding the migration metadata or as a non-modifying annotation of the process description. However, this flexibility requires the assumption of a common process consisting at least of a finite number of *activities* representing the tasks to be fulfilled during process execution, and a finite number of *variables* holding the data which is used by these activities. Each activity and each variable must have the ability to be referenced by an identifier which is unique within the process (*id*), and each process itself must have an identifier which is unique for all processes in a specified environment (*unique process id*).

Besides the identifier, the general part of the metadata includes the specification of the used *process description language* which is needed to find an available MPaaS provider which supports the specified language. Furthermore, if all suitable (mobile) process engines are currently busy, the mobile process is queued. Therefore, its *priority* tag determines whether the process should be privileged, e.g. at a higher cost.

3.1 Migration Metadata

If the resources of the initial MPaaS provider are not sufficient to finish the execution of a given process, the process can be passed on to another MPaaS provider (cp. [10]). Such process migration requires that the engine has to stop the execution of the process, document its state information and find other process engines in order to transfer the remaining process, i.e. its state and control flow to one of them. The state of the process (*process state*) and the state of each single activity (*activity state*) are represented by an element of the existing migration state model (cp. [9]). Furthermore, a set of activities can be referenced as *start-activities* to mark the next activity to be executed after process migration. This relieves executing devices from dealing with activities which have already been finished - while allowing to have multiple start-activities in case the order in which the activities have to be executed is irrelevant or the activities should be processed in parallel. This *runtime snapshot* of the migration metadata can be generated automatically, setting the process and all activities to an initial state, and is updated by participating MPaaS providers during the execution of the process.

However, the process initiator often wants to further influence the way the mobile process is executed (*user defined requirements*). If the process is going to be migrated, one of the most important questions is, where the execution of the upcoming activity should be performed. The *selection type* determines which strategy is used to assign one or more activities to a specific process engine. If the selection type is *undefined* the selection mechanisms corresponds with the strategy used in the initial approach of context-based cooperation, where process engines are picked at random until all activities of the process are successfully executed. The type *fixed participant or role* determines that a specific executing entity (e.g. a human, a process engine or a role) has to execute the process or a specified set of activities. As the next participant is often determined by the control flow logic of the process itself (as e.g. proposed by [2]), the type *variable* indicates that the required entity is described as the current value of one of the process's data variables. If no such concrete participant should be specified, but the participant should be selected as a result of a computation (e.g. picking the process engine which can execute as much of the

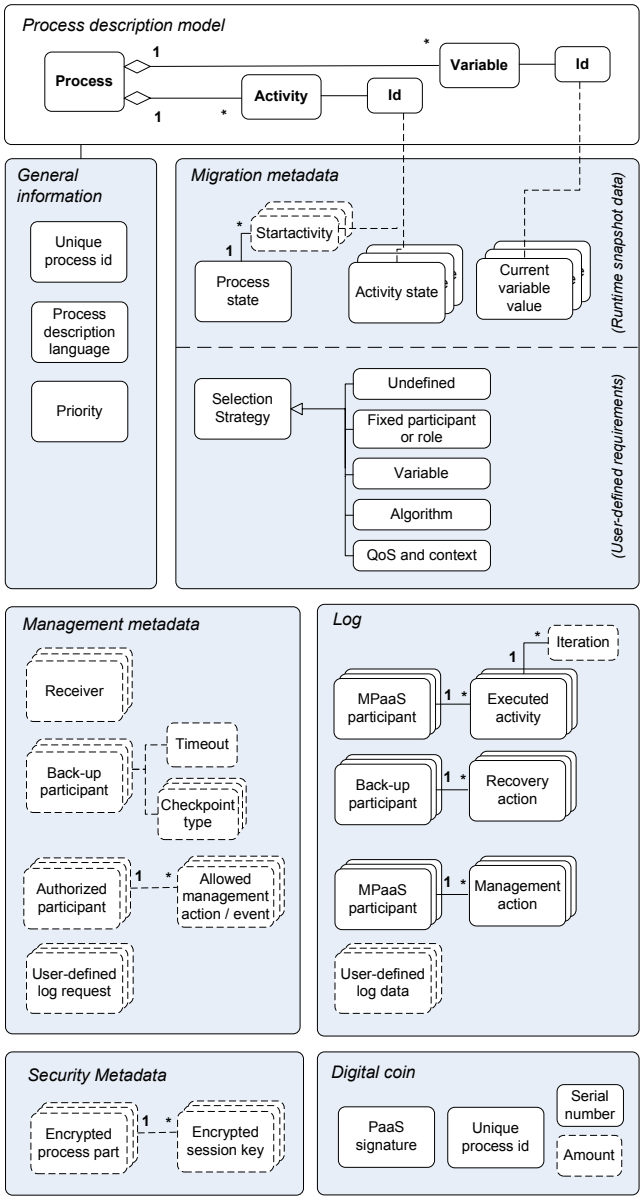

Figure 4: MPaaS metadata model

process as possible), the respective *algorithm* is referenced. Finally, the selection can rely on *quality of service and context* information such as current workload or geographical location.

3.2 Management Metadata and Log

As a consequence of delegation, the control flow of the process leaves the client's direct sphere of influence. Therefore, the management metadata represents information associated with the monitoring, logging, recovery and controlling of the distributed process execution.

First, it is important to provide sufficient management mechanisms similar to those which are utilized to control a local (non-distributed) process execution. As an example, the purchasing agent presented in Section 2.2 may want to check whether the chief purchasing officer has already started to view the proposed products - or otherwise wants to abort process execution. However, if the process migrates, it must be specified which participant should access which *management action*. However, to rather avoid unfavorable polling of state information by executing such management actions, the MPaaS consumer can additionally subscribe for a set of management events (e.g. for the completion of an activity). In addition, during and after process execution, intermediate and finale results of the process can be returned to the process initiator or to another specified participant (*receiver*).

Furthermore, in order to make mobile process execution more reliable, the MPaaS consumer can request a *back-up participant* which monitors the current MPaaS provider, i.e. the progress of the process. Therefore, the MPaaS consumer can specify one or more *checkpoint types* which determine under which circumstances a copy of the latest process metadata has to be passed to the observer (e.g. after successful execution of each activity). However, in case a specified *timeout* is exceeded, process execution is restarted on the basis of the latest checkpoint. In most cases, process execution will be monitored by the MPaaS consumer. However, also the delegation of the back-up role is possible.

An important part of the management metadata is constituted by the process's log data. In order to allow an accurate accounting based on the respective contribution, each active *MPaaS participant* is logged together with its *executed activities*. In case the activity is part of a loop, also the respective *iteration* is logged. The same is necessary in case of exceptional situations, e.g. if a process recovery has to be carried out, and in case of other management actions, e.g. an abort initiated by the MPaaS consumer. Furthermore, the process initiator is optionally able to specify which additional log data should be collected in order to allow a user-defined analysis of the process, e.g. by the collection of activity execution times or the occurrence of specific events.

3.3 Security Metadata

To exclude malicious behavior of MPaaS participants, the presented approach assumes a basic cryptographic key infrastructure, such as PKI (Public Key Infrastructure) or subject-related shared keys which are also applicable to mobile environments. Public keys and identities of participating MPaaS providers and consumers can be initiated during registration (cp. phase 1 of Figure 3).

The integrity of the MPaaS metadata (and especially its log data) can be ensured by using standard MACs (Message Authentication Codes) and digital signatures for each security-related item. However, as another important aspect, also the process description may contain private data (e.g. credit card information), private control flow information (e.g. existence of customer complaints), or identities of persons and companies which must not be revealed to or modified by other (external) parties. Considering the use case scenario, the purchasing agent e.g. has to be sure that only his supervisor executes the *confirm purchase* activity - and that the purchase is not visible to competitors or modified by malicious participants. Thus, the process modeler should be able to decide which information should be accessible for which participant. However, most process description languages (such as XPDL and WS-BPEL) allow the definition of global variables which can be referenced in several activities - and thus might belong to more than one participant. In consequence, these parts (i.e. activities and

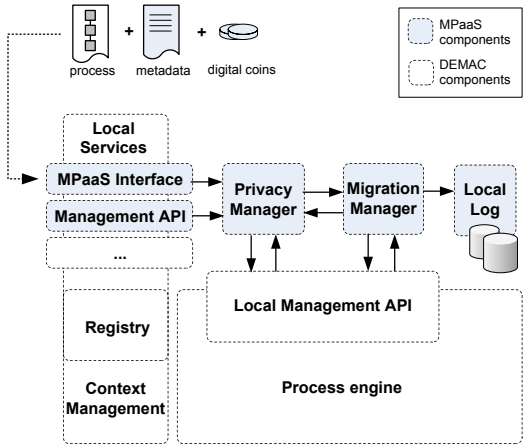

Figure 5: MPaaS provider overview

data) cannot be directly encrypted with the public key of the authorized subjects, but are encrypted with a *session key* which is only used once. Encrypted global variables can thus be accessed by different authorized subjects using the same session key (cp. [3]) which are included in the security part of the MPaaS metadata. In case of an existing PKI, the entries are encrypted with the public key of the authorized subject and can be unlocked with the respective private key. Thus, neither an additionally interaction between the process initiator and the subjects nor an authentication is required.

To additionally ensure the integrity of the process description, the process initiator is optionally able to generate a MAC for the sensitive process part. Each peer provider owning the appropriate session key is thus also able to verify the integrity respectively. However, after a participant has modified a part of the process or of the metadata it has to generate a new MAC. This possibility is indispensable because variables and metadata entries have to be changed by the subjects during process execution.

3.4 Payment Metadata

Digital coins hold a signature of the issuing PaaS provider and can optionally hold a monetary value as a pre-paid *amount*. If the MPaaS consumer initiates the execution of a mobile process, the digital coin is branded so that it can only be associated with the respective process instance (one-way coins). If necessary, digital coins can also be made anonymous in order to protect the buyer's privacy. A possible approach for this scenario is the client-side generation of anonymous coins for offline payment on the basis of group signatures (cp. [8]) – ensuring anonymity only if the consumer avoids double-spending of coins.

4. REALIZATION

The concept of context-based cooperation and the execution of mobile processes is realized by the *DEMAC (Distributed Environment for Mobility Aware Computing)* middleware. The respective prototype component supporting process migration consists of a modular process engine, a context-management system based on context federation, and a communication system responsible for event management and messaging (cp. [10]).

Based on the existing DEMAC platform, a prototype implementation of the presented MPaaS concepts has been realized (cp. Figure 5). To share their resources, all MPaaS providers have to provide a compliant interface in order to receive process definitions from the MPaaS consumer as well as from other peers. To support specialized mobile devices (e.g. devices without user interface) and mobile applications producing and initiating processes automatically, the MPaaS functionality is offered using a mobile web service architecture as presented in [15], allowing to provide and consume services based on different technological capabilities of individual mobile devices. The prototype MPaaS interface is designed to support standard web services (i.e. using HTTP, SOAP and WSDL) in order to ensure interoperability with stationary systems, as well as mobile web service technologies using protocols with less description overhead such as ASN.1 and overlay network transport (cp. [15] for details). Furthermore, the service is published in the distributed DEMAC registry as proposed in [16], so it can be found and integrated dynamically whenever a potential PaaS consumer wants to initiate an (ad-hoc) process execution. The service receives the process description and its associated migration document as input parameters and returns the unique identifier of the process and the MPaaS provider's signature in order to acknowledge its receipt.

The functional MPaaS service interface is complemented by a set of management services which are encapsulated as a lightweight version of the WSDM resource property (cp. [7]). Existing implemented management capabilities involve *status requests* as well as *suspend*, *resume* and *abort requests*. The event system supports activity events (*started, finished, failed*), process events (*started, finished*) and variable events (*variable changed*) which are subscribed by the MPaaS consumer by adding a respective entry to the metadata document.

If security mechanisms such as proposed in Section 3.3 have been applied, a simple *privacy manager* is responsible for decrypting and encrypting the process and relevant parts of the metadata. Encryption of protected process parts is realized by common procedures such as AES (Advanced Encryption Standard). To support processes described in XML syntax, the specifications *xml-encryption* and *xml-signature* by the W3C are utilized. However, concerning the "masking" of processes, it has been found that often encrypted parts of the process are causing errors during process execution because the process engine tries to interpret encrypted variables and activities but does not find expected content, e.g. encrypted variables do not meet the expected data type. Thus, the privacy manager is also responsible for exchanging non-assigned encrypted parts by temporary dummy variables or activities. As encrypted process parts are not required to actually execute the assigned parts as defined by the initiator's security policy, this does not influence process execution at the local site.

A downstream *migration manager* interprets the migration document containing migration metadata as specified in section 3.1. It is responsible for passing the given process to the process engine, to update process states, activity states and log data subsequent to execution, and, if necessary, to determine the next process participant in dependence of the given *selection type* of the upcoming activity - potentially making use of existing selection algorithms. Besides updating the process's metadata, the migration manager writes

Consumer side:	
Initiating participant:	
Supported platforms:	J2ME MIDP/PP, Java SE
Required protocols:	TCP, HTTP or DEMAC; ASN.1 or SOAP; WSDL
MPaaS components:	Client Application
JAR Size:	< 100 KByte
Back-up participant:	
Supported platforms:	J2ME MIDP/PP, Java SE
Required protocols:	(same as initiator)
MPaaS components:	Management client
JAR Size:	< 300 KByte
Provider side:	
Proxy participant:	
Supported platforms:	J2ME MIDP/PP, Java SE
MPaaS components:	MPaaS interface, subset of migration manager
JAR Size:	< 100 KByte (excl. alg.)
Full participant:	
Supported platforms:	J2ME PP, Java SE
Supported PDLs	XPDL, WS-BPEL, DPDL
MPaaS components:	all
JAR Size:	< 3 MByte

Table 1: Properties of prototype components

a local log which holds information about the contribution in the execution of external processes as well as information about which MPaaS participant has invoked the local MPaaS service and, if necessary, which MPaaS provider was selected for the continuation of process execution.

Table 1 summarizes the most important properties of the developed prototype components. To initiate process execution, the MPaaS consumer at least has to implement a suitable communication protocol and a basic service management layer (e.g. for detection and invocation of the MPaaS service). It furthermore has to store the digital coins as well as the templates for the process description and the metadata. Apart from these prerequisites, only an application-specific client is required in order to initiate the process, e.g. a graphical user interface. Additionally, the advanced MPaaS consumer (capable of acting as an observer and back-up participant) has to implement a small management client which presents a local proxy of the Management API as provided by full MPaaS participants. Considering the provider side, the proxy participants have to support the interpretation and processing of the *general metadata* and the *user-defined requirements* regarding the targeted migration of the process (cp. Figure 4). Thus, their size is generally dependent on the number and type of specific algorithms in order to compute upcoming participants. Full MPaaS participants implement the entire architecture as depicted in Figure 5, including the DEMAC middleware and at least one process engine. Besides the proprietary DEMAC process description language (DPDL), also mobility-enabled subsets of WS-BPEL (e.g. [6]) and XPDL can be supported.

5. RELATED WORK

Initial approaches in the area of PaaS either rely on a single powerful process management system or on a centralized cluster of process engines which knows and controls all system resources. Although the need for mobile participants has already been recognized, most often a stable network infrastructure and a ubiquitous accessibility of resources and services are assumed. Thus, existing commercial PaaS solutions already allow to involve mobile human process participants, e.g. by web browser, e-mail or sms [4], but nevertheless, the integration of decentralized mobile process management systems as described above is still an open issue.

Relevant approaches in the area of mobile process management systems and their applicability are summarized in Table 2: The monolithic mobile process engine *Sliver* [6] is able to execute a subset of standard WS-BPEL processes on a mobile device by invoking standard web services running on stationary servers or on the mobile device itself. As (sub)processes are not allowed to leave the system on which they have been initiated, the process does not need and thus does not implement any additional distribution or security mechanisms. As a hybrid approach, the *Exotica/FMDC* workflow management system [1] enables mobile clients to download single user tasks or simple sequential activity blocks in order to perform them while temporarily being disconnected from the central workflow server. However, decentralized execution and sharing of resources are not supported here. As a more cooperative approach, the *WORKPAD* infrastructure [11] was designed to support human rescue teams in disaster scenarios, but still requires a central entity in order to coordinate mobile process participants at runtime. In contrast, a choreography-based workflow management system targeted at mobile environments is represented by *CiAN* [13] which supports distribution on the basis of process fragmentation. As the overall process is physically cut into process fragments which are distributed to dedicated mobile process engines, privacy of critical process parts is not a problem. However, choreographies and process fragmentation need a joint preparation phase for the physical distribution of each business process, where all participating parties have to be available. Thus this approach is more advantageous in case of the recurrent execution of the same process than for unfrequent ad-hoc processes. Similarly, the approach of *MobiWork* [5] realizes mobile workflows for ad-hoc networks and is focused on the allocation of tasks to mobile participants also using process fragmentation to generate "sub-plans". As introduced, the *DEMAC* process management system [16] is able to delegate process execution (in whole or in part) to other mobile or stationary process engines using the concept of context-aware process migration.

As shown in Table 2, the presented approach of MPaaS is complementary to existing (mobile) process management systems. The DEMAC approach of runtime delegation is most closely related and forms the basis of the distributed mobile PaaS system. However, other systems which focus on the execution of an individual process description language, e.g. Sliver, can even be integrated in order to share their functionality, e.g. the execution of WS-BPEL processes.

6. CONCLUSION

This paper proposes a mobile PaaS solution to flexibly share mobile and stationary process engines in order to enable execution of mobile processes even without the availability of an one-for-all central PaaS server. The presented approach advances the concept of context-based cooperation to support the user-defined migration and execution of ex-

Requirement:	Sliver	Exotica	Cian	WORKPAD	MobiWork	DEMAC	MPaaS
R_1 : Sharing of external resources	-	-	+	+	+	+	
R_2 : Payment and accounting	-	-	-	-	-	-	+
R_3 : Process privacy and security	n/a	-	(+)	-	(+)	-	+
R_4 : Traceability	n/a	n/a	-	-	-	-	+
R_5 : User-defined execution	n/a	n/a	-	-	-	o	+
R_6 : Decentralization	-	-	o	-	o	+	
R_7 : Context awareness and QoS	-	-	o	o	o	+	
R_8 : Dynamic discovery	-	+	-	-	+	+	

Table 2: Analysis and comparison of related approaches

isting process models and proposes a management concept as well as a basic accounting model. Regarding the identified requirements, it can be concluded that the concept of context-based cooperation realized in the DEMAC project together with the presented MPaaS approach fulfills most of the given requirements (cp. Table 2).

However, a key success factor of the MPaaS system is the number and geographic distribution of registered private peers and thus their availability, assuming that at least one suitable provider is available at the MPaaS consumer's site. If all (mobile) participants are both MPaaS consumers and peer providers, the effectiveness of the system scales with an increasing number of participants. More potential consumers promise more profit for peer providers and thus the willingness to share resources will increase. Moreover, business processes initiated in mobile environments are often only marginally related to the sensitive core business logic, such as the process presented in Section 2.2. For these reasons, both the necessity and willingness to cooperate is much higher than in traditional stationary systems, but in contrast also requires a critical mass of participating systems. Thus, the proposed concepts still have to be evaluated in a broader study on the acceptance of PaaS.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

8. REFERENCES

[1] G. Alonso et al. Exotica/FMDC: Handling disconnected clients in a workflow management system. In *Cooperative Information Systems*, pages 99–110, 1995.

[2] T. Bauer and P. Dadam. Efficient Distributed Workflow Management Based on Variable Server Assignments. In *CAiSE 2000*, pages 94–109, 2000.

[3] E. Bertino, S. Castano, and E. Ferrari. Securing XML documents with Author-X. *Internet Computing*, 5(3):21–31, 2001.

[4] P. Fingar. *Dot Cloud: The 21st Century Business Platform Built on Cloud Computing*. Meghan-Kiffer Press, 2009.

[5] G. Hackmann et al. MobiWork: Mobile Workflow for MANETs. Technical report, Washington Univ., 2006.

[6] G. Hackmann et al. Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. In *Int. Conf. on Service-Oriented Computing (ICSOC 2006)*, pages 503–508. Springer, 2006.

[7] W. V. Heather Kreger, Vaughn Bullard. Web Services Distributed Management: Management Using Web Services. Technical report, OASIS, 2006.

[8] X. Hou and C. H. Tan. A New Electronic Cash Model. In *Proc. of Int. Conf. on Information Technology: Coding and Computing (ITCC 2005)*, pages 374–379. IEEE, 2005.

[9] C. P. Kunze, S. Zaplata, and W. Lamersdorf. Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments. *Journal of Computers*, 2(1):1–11, 2 2007.

[10] C. P. Kunze, S. Zaplata, M. Turjalei, and W. Lamersdorf. Enabling Context-based Cooperation: A Generic Context Model and Management System. In *11th Int. Conf. on Business Information Systems (BIS 2008)*, pages 459–470. Springer, 2008.

[11] M. Mecella et al. WORKPAD: An Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios. In *Int. Symp. on Collaborative Technologies and Systems (CTS'06)*, pages 173–180. IEEE, 2006.

[12] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *15th ACM Symposium on Principles of Distributed Computing*, 1996.

[13] R. Sen, G.-C. Roman, and C. D. Gill. CiAN: A Workflow Engine for MANETs. In *COORDINATION 2008*, pages 280–295. Springer, 2008.

[14] M. Turner, D. Budgen, and P. Brereton. Turning Software into a Service. *Computer*, 36(10):38–44, 2003.

[15] S. Zaplata, V. Dreiling, and W. Lamersdorf. Realizing mobile web services for dynamic applications. In *9th IFIP Conf. on e-Business, e-Services, and e-Society (I3E 2009)*. Springer, 9 2009.

[16] S. Zaplata, C. P. Kunze, and W. Lamersdorf. Context-based Cooperation in Mobile Business Environments: Managing the Distributed Execution of Mobile Processes. *Business and Information Systems Engineering (BISE)*, 2009(4), 10 2009.