# Enabling Context-based Cooperation: A Generic Context Model and Management System

Christian P. Kunze, Sonja Zaplata, Mirwais Turjalei, and Winfried Lamersdorf

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[kunze|zaplata|9turjalei|lamersdorf]@informatik.uni-hamburg.de

**Abstract.** In order to realise complex service-based applications on system platforms for context-aware ubiquitous computing environments, *mobile processes* have been introduced to support cooperation among (mobile) devices by exchanging and executing arbitrary (business) processes. In such a view, middleware platforms that support the execution and migration of mostly a priori unknown processes need a generic and also application-independent context system. Accordingly, this paper presents an approach for a generic context model and management platform to support such *context-based cooperation* as currently developed and used in the project DEMAC (*Distributed Environment for Mobility-Aware Computing*).

## 1 Introduction

Mainly due to the technical progress in processor and network technology, ubiquitous computing environments are becoming more and more reality. One of the main characteristic of such environments is the mobility of users, devices, and even application code. Under such conditions, mobile applications and supporting platforms frequently have to bridge the gap between required and provided resources and capabilities at any specific place and/or time.

In order to narrow this gap, devices of respective mobile vicinities can *cooperate* by sharing their resources for executing mobile applications. Today however, in most cases, mobile applications are still restricted to the capabilities of those device(s) they were initiated on. Other resources which are, e.g., potentially available from other devices remain still inaccessible for any dynamic adjustment of the mobile application. In consequence, this also limits the complexity of applications and tasks to the initialing device's capabilities. But, in order to fully realise the vision of ubiquitous computing [16], even much more complex and also unknown tasks (and thus more generality) have to be supported by advanced mobile and context-aware systems. *Mobile processes* represent such complex application tasks and use context knowledge to execute and migrate (business) processes in order to increase the likelihood to finish the task successfully by integrating the capabilities of different mobile nodes of the vicinity

(cp. [8] for a detailed introduction). As use of these processes leads to real cooperation among the participating mobile systems, applications implemented as mobile processes are able to realise a form of *context-based cooperation* [9]. However, a supporting context-aware middleware for this new class of applications has increased requirements for the underpinning context model and management system as compared to more traditional ones. Accordingly, this paper identifies such requirements and presents an approach to model and use context for a mobile system supporting the execution and migration of mobile processes.

The following subsections of the paper first motivate the need for a generic context model and management system for mobile processes. Afterwards, basic requirements of an underpinning context-aware middleware platform to support such processes are identified. Section 2 then addresses related work and section 3 presents the coarse architecture, the proposed context model and the corresponding context management component. Section 4 refers to the evaluation within the DEMAC middleware before section 5 concludes the paper with a summary.

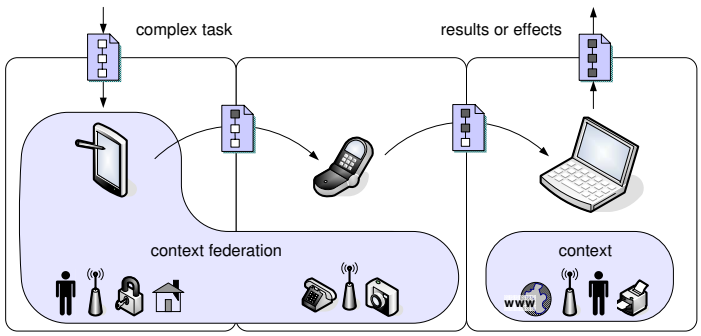



**Figure 1.** Context-based Cooperation for Mobile Processes

### 1.1 Executing Service Compositions with Mobile Processes

A *mobile process* is a goal-oriented composition of arbitrary services and manual tasks which may span several heterogeneous mobile and static devices, users, and services. In order to (potentially) use all the capabilities and resources provided in its entire local and remote environment, such a process can *migrate* to other devices, e.g. to share the functionality provided by these nodes. In order to ensure the user's goals even on foreign devices, non-functional requirements can be specified to restrict participating services, users and devices. The required activities within such processes are determined by abstract service classes to refer and identify applications and services in a technology-independent way. This is done because the type of services and applications provided in the environment at runtime can not be determined at designtime - but result from the available

resources of the context during execution. The same argument holds for the description and the management of context information: Because a mobile process can involve arbitrary tasks and heterogeneous devices - and therefore also different contexts which cannot be determined in advance by the executing device - the context model and context management system of a supporting middleware infrastructure has to be generic and application-independent even at runtime. Figure 1 shows an (abstract) mobile process migrating three devices in dependence of the discovered context. As long as the process engine of a device is able to bind local or remote services to its current activity, it is responsible for the mobile process. However, in cases of failures or lack of respective service instances the engine has to try to find other devices to execute the mobile process and to transfer the remaining process and its execution to one of them. As the initiator of the process (in this case the user of the PDA) is interested only in the effects of the process execution, there is no need to return the results to this participant. A concrete example of such a mobile process can be found in [9].

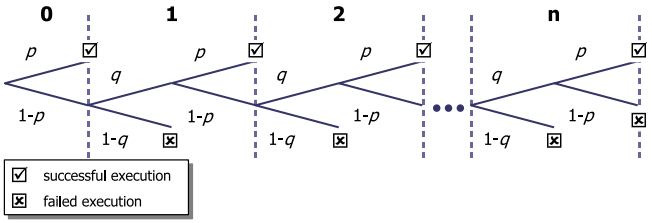
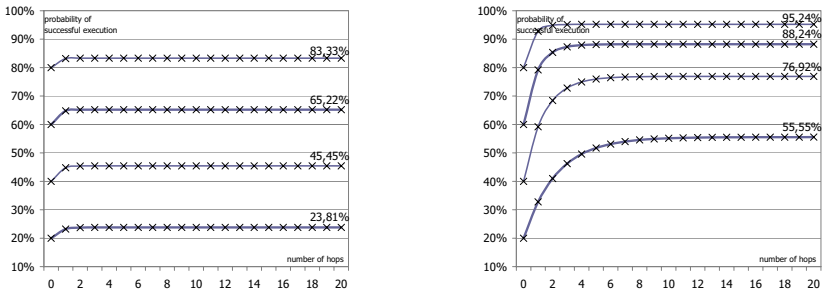



**Figure 2.** Probability tree of successful execution

The procedure of migration opens up a new vicinity to search for other and maybe more suitable devices and is determined by the heterogeneity of the vicinity. Thus, the likelihood to finish the overall task successfully is increased in the following way (as depicted in figure 2): The upcoming task can either be directly executed by the current device itself or, otherwise, has to migrate to another device. Accordingly, let $p$ denote the probability of a single device being capable of executing the current task. In extension, let $q$ denote the probability of migration. Without restricting generality, let furthermore $p$ and $q$ be equal for all devices of the mobile vicinity, and $n$ be the number of hops caused by migration. Equation 1 summarizes these observations by calculating the probability as a converging geometric series as the likelihood of a successful execution of the task *anywhere* in the mobile vicinity. Some exemplary values are presented in figure 3 showing the probabilities of successful execution with a migration probability of 20% and 80%. As to see, the estimated probability of a successful execution increases considerably already after only a few number of hops.

$$P_{Success} = p\sum_{i=0}^{n}((1-p)q)^i = p\frac{1-((1-p)q)^{n+1}}{1-((1-p)q)} \tag{1}$$




(a) Migration (q=20%) (b) Migration (q=80%)

**Figure 3.** Estimated probability of successful execution

In summary, the probability of successful 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, since such different context cannot be determined in advance, the context model and context management system of a supporting middleware infrastructure has to be *generic* and *application-independent* even at runtime – which is hardly supported by traditional context-aware middleware systems. The following section therefore presents basic requirements for such a generic context-aware middleware approach for mobile processes.

### 1.2 Requirements Analysis

In order to facilitate the exchange of context information and interaction between heterogeneous mobile devices, relevant *context information* has to be mapped to an adequate data structure and thus be provided in a standardized way (1). Therefore, on the one hand, context information has to be represented by an adequate *uniform and extensible model* (2) which allows standardized interaction between mobile devices. On the other hand, techniques for context management, such as the discovery, evaluation and administration of relevant data, are required. To ensure reusability and extensibility, the context model and the context management system should be *conceptually decoupled* (3). Additionally, the entire context system has to be independent of platforms, communication protocols and programming languages to allow a *flexible integration* into existing mobile middleware (4).

A main aspect of cooperation is the *exchange of distributed context information* (5). Thus, not only the local context but also the contexts of other mobile participants in the vicinity are relevant. Therefore, the middleware should define a protocol to exchange context information and to provide efficient mechanisms to access the context of cooperating devices. Because different devices can include different contextual data, the context model has to be able to represent different categories of context information (e.g. environment, activity, identity or time). Therefore, it is necessary to *abstract* from the structure of contexts and

their actual representation, e.g. the local time, temperature or velocity (6). Consequently, the management system has to support the *transformation* of context information to equivalent data formats on the basis of semantic knowledge (7).

In order to enable proactive and ad-hoc reactions to relevant changes of context, the middleware needs to observe the context model and *notify registered applications in case of specified changes* (8) in the overall context or of single attributes. Other important requirements are derived from considering constraints resulting from mobility. The scarcity of resources of mobile devices requires a *lightweight and robust system architecture* (9). This means, in particular, that the composition of middleware services has to be *flexible and scalable* (10) in order to allow, for example, deactivation of temporarily unused modules. Likewise, the selection of relevant and situation-specific context information requires custom *filter mechanisms* (11). Concerning the context model, the semantic separation of *high-level, low-level, dynamic and static context information* facilitates the runtime management (12): While dynamic context data is subject to frequent changes and has to be updated periodically, static context does not need costly refreshing procedures at runtime. Furthermore, non-functional aspects such as *security and privacy concerns* (13) as well as the specification of quality parameters (14) for the potentially imprecise data play an important role (*Quality of Context* [2]).

## 2 Related Work

As shown, context-based cooperation of services requires a generic context model as well as a flexible management system to support arbitrary applications defined as a mobile process. Respectively, this section presents an overview of related context modelling approaches and existing frameworks to support context management on mobile systems. A rather simple and thus lightweight approach of structuring context information is realized by the *key value model* [13]. To provide context information, the value of the relevant context parameter - like the current position - can be stored e.g. to the key of one respective environment variable [12]. As a consequence of its simple representation, this model is not adequate to structure composed or high-level context information. In contrast to that, *markup schema models* represent context data in more complex hierarchical textual structures [13]. Examples are the *Comprehensive Structured Context Profile (CSCP)* [6] and the *CC/PP Context Extension* [7]. Related to this markup approach, an *object oriented model* encapsulates context information also hierarchical but as classes. A more formal description of context can be derived by *contextual reasoning or inferencing* in logic-based approaches [10] or can be realized by *ontological models*, which introduce methodologies for a normalized domain-specific knowledge representation. Examples are *Aspect Scale Context Information (ASC)* [10] and *CONtext ONtology (CONON)* model [15].

These models are used, e.g. in the *Context Toolkit* [11] which is a management framework based on an object-oriented architecture. It uses *context widgets* to abstract from sensor-based raw data, provides reusable components to

access context information, and supports the exchange, persistence and logging of context data. Although the widget approach is a suitable solution to abstract from low-level in-formation and to provide reusable components, there are no concepts about how to exchange context information in a distributed mobile environment. In contrast, the *Hydrogen* framework [14] was exclusively developed for mobile applications and is therefore characterized by a very lightweight architecture. However, communication between applications, middleware and sensor network is restricted to TCP/IP-based protocols. Furthermore, a transformation mechanism to obtain different but equivalent representations of context information is not available. The *Java Context Awareness Framework (JCAF)* [1] is a compact event-based system to support the development of context-aware platforms. It consists of an extensible API to model application-specific context information, and of a distributed infrastructure system, which provides cooperating context services in order to manage the context model. However, a clear separation between context model and management system is not consequently pursued. Furthermore, the communication between services and clients as well as among the clients themselves is based on Java RMI and, in addition, context information is modelled as serializable Java objects and thus JCAF cannot be considered to be totally platform-independent.

| | **Context Toolkit** | **Hydrogen** | **JCAF** | **DEMAC Approach** |
|---|---|---|---|---|
| (1) Provision of Context Information | + | + + | + | ++ |
| (2) Extensibility and Reusability | + + | + | − | ++ |
| (3) Separate Model and Management | − − | − | + | ++ |
| (4) Integration | + | + | − | + |
| (5) Distribution | − | + | + | ++ |
| (6) Mapping and Abstraction Level | + | − | + | + |
| (7) Transformation | + | − − | + | ++ |
| (8) Extended Notification Mechanism | − | − | − | + |
| (9)Robustness and Compactness | + | + + | + | ++ |
| (10) Flexibility and Scalability | − − | − | − | ++ |
| (11) Filter Mechanisms | − | − | − | ++ |
| (12) High Level and Static Context | + | − | − − | ++ |
| (13) Security and Privacy | − − | − | + | − |
| (14) Quality of Context Parameter | − | − | + + | ++ |

++ supported; + partly supported; − marginally supported; − − not supported

**Table 1.** Analysis of Context-Aware Middleware Platforms

Although a detailed discussion of the whole range of related work is out of the scope of this paper, in general, it can be derived that *peer-to-peer* and *ad-hoc* infrastructures of portable devices form the most challenging environment for mobile applications and, consequently, systems which move at least important parts

of the context computation into supporting infrastructure - such as *Nexus* [4] or *Solar* [5] - and relatively heavyweight ontological approaches - such as *SOCAM* [15] or *CoBrAs* [3] - are not considered in this paper. Table 1 evaluates (due to space restrictions only some) selected contemporary context-aware middleware approaches and the proposed system according to requirements for supporting context-based collaboration and outlines also exemplary deficiencies. In particular, the evaluated approaches 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 and to realize extended notification to consider changes in the overall context remain unsupported. Therefore, all approaches presented there do not suffice to support highly dynamic context-based collaborations which can also determine relevant context parameters at runtime.

## 3 A Context Component for Distributed Mobile Systems

In line with the well-established tradition of "middleware-based" approaches to open distributed application development, system support for mobile, context-aware applications shall also be realised by respective *infrastructure components*. In such a view, figure 4 provides an overview of the coarse architecture of a *context (middleware) component*. As shown, the *raw data layer* represents a collection of heterogeneous services and components which provide low-level context information without any structuring. These are either interfaces to physical sensors of the mobile device itself or logical sensors based on (remote) services. As the components of the raw data layer are composed arbitrarily and can also change during runtime, context information can be achieved from different resources. Therefore, the *service management* of the *context layer* has to abstract from these single resources in order to overcome the heterogeneity of its underlying services. The context management component uses homogeneous wrapper services from the service management to administer the context model, e.g. to update context data on the basis of new raw data. Both components are encapsulated by the *context service*, which acts as a proxy to access the context layer.

### 3.1 Generic Context Model

Although resources and services providing context are often heterogeneous and differ in representation and quality, the middleware should be able to allow for an intuitive and non-complex selection and composition of context information. Therefore, the context model has to provide standardized interfaces and formats to integrate different context resources. To illustrate the components of the presented context model and the inter-relationships, figure 5 shows a schematic diagram of this model: Here, an entity represents a single object which can aggregate several components to describe contextual information on a higher level as, e.g. a person with her location and activity or a meeting combining a set

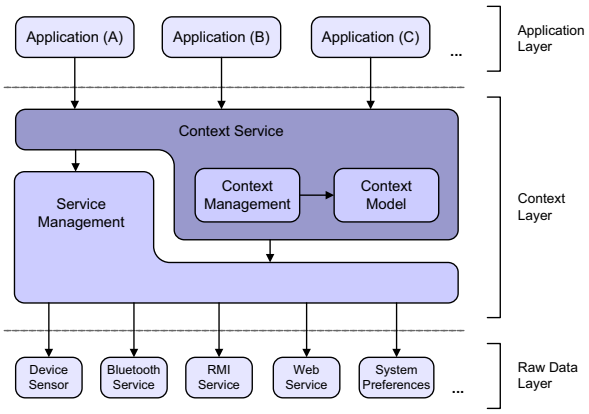



**Figure 4.** A Context Component to Support Context Aware Middleware Systems

of persons. As can be seen in this example, complex entities can also aggregate other entities to describe their context recursively.

A single piece of context information within an entity is modelled as an *attribute*. Because the context information can be obtained by heterogeneous services with different characteristics, there are also different types of attributes. A *sensed attribute* indicates a dynamic context value, meaning that the context value changes relatively frequently which is typical for information from logical or physical sensors (e.g. the amount of free memory space of a mobile device). In contrast, a *defined attribute* models more static context information, which is unlikely to change during runtime, e.g. the identity of the user or the type and model of the mobile device. Finally, high-level context information, which is obtained by a combination or processing of several low-level context attributes, is described as a *deduced attribute*. The actual value of an attribute is modelled as the *data value* component and can include arbitrary simple or complex data types. Furthermore, each attribute provides a link to external semantic resources - e.g. an RDF file - as a unique identifier allowing to share context data between manifold applications more easily. Because contextual information can be imprecise, inconsistent, or incomplete, information about the quality of context helps to interpret the reliability of the data. Therefore, the component *quality constraints* contains a set of *quality parameters* as, for example, the refresh period of data, its precision, or its standard deviation.

On a higher level, the component *domain context* 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.

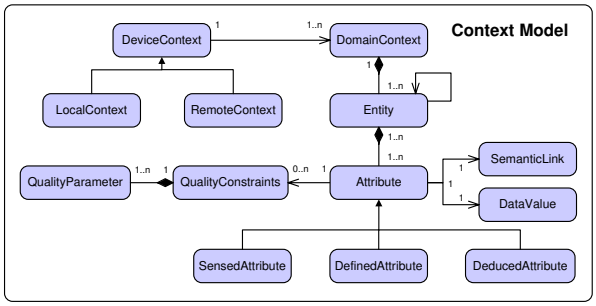



**Figure 5.** Schematic Diagram of the Context Model

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 next to the *local context* also the *remote context* of foreign (but locally reachable) mobile systems. As a result of the generic model, the remote context can contain arbitrary domains and entities, unaffected by the (limited) set of contextual information modelled within the local context.

### 3.2 Context Management System

In order to achieve the required separation between context model and management, all middleware functions concerning the management of the presented context model are realized by autonomous services: The central component of the context management is the *RemoteContextService* which manages the exchange of context information between cooperating devices. It implements a protocol to provide local context to other devices and to integrate remote context to learn about the environment of foreign participants. Additionally, a *SecurityPrivacyService* determines which information should be revealed to other participants. A *SerializationService* is responsible for serialization and deserialization of context information by, e.g. representing the model in XML format or others. If context data is required to have a different representation, the *TransformationService* supports the conversion of contextual data values to equivalent formats. Refreshing data values of the context model, which is especially relevant for dynamically sensed attributes, is done by the *UpdateService* which obtains recent raw data from sensors and external services. A *NotificationService* observes the local context model and the relevant context of remote systems in order to allow proactive actions through event notifications. Finally, a *StorageService* saves context information in order to analyse past values and to make predictions about future contexts.

## 4 Evaluation

In order to evaluate the applicability of the generic context model and management system as proposed above, a prototype implementation of all presented components has been realized within the existing DEMAC middleware which provides basic support for the exchange and distributed execution of mobile processes (cp. [8]) in heterogeneous open system environements. For testing the integrated approach, a distributed example application has been implemented based on a set of mobile processes, each modelling a user-centric workflow to execute a sequence of manual tasks and automatic services. The mobile processes have been modelled with the *DEMAC process description language (DPDL)* (cp. [9]) and contain several activities whose execution is constrained by different contextual criteria, depending on the user's individual requirements as well as on the capabilities of the environment the process is migrating to.

The prototype implementation of the DEMAC middleware runs on different mobile devices, such as a notebook or a PDA as well as on Desktop PCs. Each collaborating device holds its own *local context model* and can obtain relevant *remote context models* if needed. Each local context includes a *DefaultDomain* with a *DeviceEntity* and either a *UserEntity* or a *CommunicationEntity* and a couple of basic context attributes, like static ones (e.g. user identity, display resolution and operating system) and dynamic attributes (e.g. available network connections, free disk space and current location). Furthermore, different services (either locally or remotely accessible) have been realised to execute upcoming (sub-) tasks of the mobile process.

Since context attributes can be exchanged and used during runtime, the processes' propagation can be controlled depending on relevant context data. This hinders mobile process to be transferred to devices outside the right context - thus avoiding unnecessary migrations. For example, in the prototype implementation, the middleware's context service is able to consider the availability of a specific participant (user or device), appropriate services to perform an upcoming task, or the compliance to non-functional requirements like network quality or memory resources. In case the context service cannot find appropriate participants with an acceptable context, the mobile process can be transferred to an arbitrary participant in order to enter a new execution environment and, thus, new contexts.

To verify the results obtained by the analytical model (c.p. section 1.1) and to analyze the applicability of the context service, the evaluation includes an experiment to determine whether or not the probability of successful execution increases by context-based cooperation based on the developed context model. The environment for the experimental series consists of a simple process with one single activity, six heterogeneous devices (as described above) with two devices having the capability to execute the processes' activity and four devices being unable to do so. Because sender and receiver of the mobile process must not be the same, there are 5 possibilities for each process to migrate from one device to another, which makes an execution probability of p=40% within the entire system. To test the behaviour of the prototype under load, three test

runs were carried out, each including a number of 100 processes. Figure 6 shows the average number of hops resulting from migration necessary to execute the process successfully. 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 probability estimated in the analytical model and the applicability of the context model can therefore be confirmed also by practical experimentation.

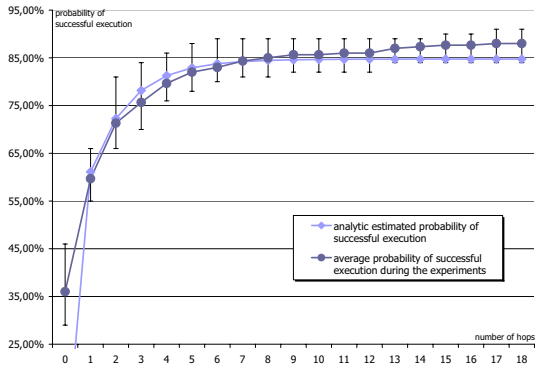



**Figure 6.** Experimental Series: Average Probability of Successful Execution

Nevertheless, a known issue of exchanging context data when no suitable participant can be found is the low performance of retrieving remote context information due to the necessity to collect it in recurrent intervals. Besides, better process routing could be achieved, if context data could be shared transitively among participating devices. Furthermore, not all relevant security issues could be resolved so far.

## 5 Conclusion and Future Work

This paper argues that *mobile processes* can be used to realize a class of context-based systems which use context information to support distributed cooperation by joint use of all combined capabilities of all different (mobile) devices in a distributed environment. Since established context models and management systems do not suffice to support such applications, requirements for middleware support for such a class of *context-based cooperation* were identified. A unified approach of an *adjusted and contemporary context model* and corresponding *management component for mobile systems* were presented and, also based on a respective prototype implementation, evaluated. Future work includes solutions for still open security issues as well as the development of advanced strategies to increase migration performance.

## References


1. J. E. Bardram. The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In *Pervasive*, pages 98–115, 2005.
2. J. Baus, A. Krüger, and W. Wahlster. A resource-adaptive mobile navigation system. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 15–22, 2002.
3. H. Chen, T. Finin, and A. Joshi. An Intelligent Broker for Context-Aware Systems. *Adjunct Proceedings of Ubicomp 2003*, pages 183–184, 2003.
4. F. Dürr et al. Nexus – A Platform for Context-Aware Applications. In *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, 2004.
5. G. Chen and D. Kotz. Solar: A pervasive computing infrastructure for context-aware mobile applications. Technical report, 2002.
6. A. Held, S. Buchholz, and A. Schill. Modeling of Context Information for Pervasive Computing Applications. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002)*, 2002.
7. J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in Using CC/PP in Context-Aware Systems. In *4th International Conference on Mobile Data Management (MDM)*, volume 2574 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 2003.
8. C. P. Kunze, S. Zaplata, and W. Lamersdorf. Mobile Process Description and Execution. In *Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2006)*, pages 32–47. Springer, 2006.
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. ISSN : 1796-203X.
10. J. McCarthy and S. Buvač. Formalizing Context (Expanded Notes). In *Computing Natural Language*, volume 81 of *CSLI*, pages 13–50. Stanford University, 1998.
11. D. Salber, A. K. Dey, and G. D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of CHI'99*, 1999.
12. B. N. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proceedings of the* $1^{st}$ *International Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.
13. T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*, 2004.
14. T. Hofer et al. Context-Awareness on Mobile Devices - the Hydrogen Approach. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, pages 292–302, 2003.
15. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology Based Context Modeling and Reasoning using OWL. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, page 18, 2004.
16. M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, 256(3):94–104, 1991.