A UML Profile for Modeling Mobile Information Systems

Vegard Dehlen and Jan Øyvind Aagedal

SINTEF ICT, Cooperative and Trusted Systems, Forskningsveien 1, 0314 Oslo, Norway
{vegard.dehlen, jan.aagedal}@sintef.no

Abstract. In this paper we propose a framework for modeling mobile information systems. Mobility introduces several challenges and issues that impact the development of mobile systems. As a result, we want applications running on mobile devices to exhibit certain traits; they should be aware of the mobility and be adaptive to the changes that occur due to it. Literature has identified several types of mobility – among them, physical and logical mobility. The former pertains to tangible mobile entities like cars, devices and people, while the latter encompasses mobile software entities. In addition to these, this paper includes the concept of *vertical mobility* – the movement of a network connection between overlapping networks – in a UML profile for modeling mobile information systems. We discuss our experiences from a case study described in [1] , where we modeled a simple mobile information system and transformed parts of the model into code.

Keywords. Mobility, UML profile, model-driven development.

1 Introduction

The introduction of small hand held devices with Internet connection is rapidly changing the way we both work and live, and an increasing number of people are acquiring these devices. In today's society we can identify several mobile devices. Laptop computers, cell phones, PDAs and tablet PCs are all examples of devices that can be used while moving around. Common usage is accessing e-mail, remote databases or the Web, sending faxes and making phone calls, scheduling and document processing [2], in addition to newer usage areas like watching TV or movies, performing video phone calls or downloading music.

The emergence of novel and useful services and applications in a domain is highly dependent on software engineering. The existence of a solid development framework and methodology allows applications to be developed more rapidly and with higher quality, in addition to promote consistency, interoperability and reuse within the community. Such a framework should capture the characteristics and concepts of the target domain. The work in this paper builds upon and expands the previous efforts towards reaching this goal, i.e., representing the mobility domain at the metalevel.

This paper is organized as follows. Chapter 2 gives an analysis of the problem, by analyzing the concept of mobility and what we mean by it and by introducing differ-

ent types of mobility. In Chapter 3 we list some requirements for a framework for modeling mobile information systems. Chapter 4 presents our solution to the problem; a mobility metamodel and an accompanying UML profile. Chapter 5 discusses the validity of our solution, before Chapter 6 draws some conclusions and suggests future work.

2 Problem Analysis

2.1 Theory of Mobility

Since its inception, mobile computing has resulted in the introduction of several subfields, and today we talk about systems that are context-aware, location-aware, mobility-aware and/or adaptive. In the following, we will further explain our views of mobility and the kind of applications we are interested in modeling.

An adaptive system simply refers to a system with the ability to adapt to different situations and contexts. Adaptation is not a phenomenon exclusive to mobile computing, but it is, as pointed out earlier, identified as the main strategy for coping with the high variability and heterogeneity of the mobile domain [3, 4]. There are several factors an application can adapt to:

1. To its current context or changes in context.
2. To its available system and network resources and changes in these.
3. To changes in location, i.e., mobility.

There are different ways to view mobility. First, we can see it as an entity's ability or willingness to move. Second, we can see it as an entity that is currently moving. Movement patterns can be described by different modalities, as defined in [7]. Third, we can view mobility as a change of an entity's location, where the movement between locations is considered an atomic action.

In the area of context awareness, change of location is interesting due to the changes in context that naturally occur. Location change might entail changes in several environment properties like temperature, nearby people and devices, available printers and ongoing activity - properties a context aware application can take advantage of. In the field of mobility, location change also means roaming between different network cells, requiring seamless handoff and service [8] and session [9] mobility. Some of these issues are handled in the network and middleware layers, but the application can also take advantage of these activities, like employing a new mobile code strategy based on the change in network characteristics. However, in an adaptivity context, these issues alone are not enough to warrant the concept of location as a first class entity, as an application does not need to know of any other than its current location to perceive changes in context and network resources. A system that only considers location change can offer *reactive* adaptation, which means that it can react to the changes that occur because of mobility.

Modeling locations as first class entities is only necessary in an application that needs to know the properties of locations other than its current one, which is enabled by an entity's ability to move. A system can then provide *proactive* adaptation.

There are different ways of representing a location. The abstraction we choose depends on the unit of mobility, where location could be represented by Cartesian coordinates for a mobile device or by a host address for a mobile agent [6]. Optionally, we can choose abstractions that are conceptually related to the world we live in, where a mobile device could be located in the tax free shop at Gardermoen airport in Norway. In the latter scenario, we see that locations are defined within locations. In addition, locations can be mobile. A passenger on a ferry will have a location relative to the boat (being in his cabin, for example), while the boat has a location relative to its previous and destination port. In addition to being nested, locations can be overlapping. An example of this is a road that runs through several areas of a city. One could thus say that, conceptually, an entity has two different locations. However, in practice, we consider the entity to be located in the intersection of the overlapping locations.

Another reason for treating locations as a first class concept is, as identified in [11], that locations may have access restrictions or barriers. A person traveling from one country to another will have to pass security mechanisms at the border, while a mobile agent might have to pass a firewall to access a remote device or administrative domain. These concerns are out of this paper's scope, but the concepts of mobility and locations presented provide a foundation on which security and access control can be modeled and reflected upon.

2.2 Types of Mobility

Physical and Logical Mobility. Literature has long since identified two main types of mobility. Logical mobility (also called mobile computation) deals with the movement of software entities, while physical mobility (mobile computing) deals with the movement of physical entities.

There is a distinct difference between physical and logical mobility. The former is something that occurs in the real world, as people or devices move and change locations. Each location might offer different resources and context, like nearby printers or available networks. An application running on a mobile device might thus continually experience change in available resources and context. A mobile information system cannot control or influence physical mobility, but it can observe location changes and react with different adaptation strategies if necessary. For logical mobility, on the other hand, the situation is the total opposite, as logical mobility is a phenomenon that encompasses software entities that are designed by an application developer. Consequently, while an application reacts to physical mobility, it can employ logical mobility and mobile code as an adaptation strategy – possibly as a reaction to physical mobility.

It is worth noting, however, that logical mobility can exist without physical mobility and vice versa.

These fields are mostly disconnected; logical mobility within the software community and physical mobility within the hardware community. However, [11] argues that the two types of mobility are intertwined, and should be treated in a uniform way.

Vertical Mobility. As time progresses, more and better access points become available in our environment. Especially in high density areas, a device can have several heterogeneous access networks to choose from. These networks might offer different services, coverage, cost and bandwidth, and the mobile device can choose which network to use. Change of access network is thus not only caused by physical mobility, but might also happen while the device remains stationary. This is called vertical handoff. The term vertical refers to overlapping wireless networks and their hierarchical and asymmetric relationship [12]. A device can thus have access to networks that offer low-bandwidth over a wide geographic area to networks that offer high-bandwidth over a narrow geographic area [13]. The opposite is called horizontal handoff, where the handoff occurs between access points in a homogeneous network infrastructure [14]. An example of horizontal handoff is when a mobile phone switches between different access points.

One of the main problems of mobile systems is that a mobile device will have to change access network, which can be divided into three different scenarios:

1. The device leaves the coverage area of its current network, and loses connection.
2. The device leaves the coverage area of its current network, and connects to another available network.
3. The device is stationary, and chooses to connect to another available network.

The first two scenarios are direct results of physical mobility, where mobile nodes move out of their present network coverage. The third scenario, however, is not true mobility, but has the same effect; the system must manage the change in IP routing caused by the vertical handoff [15]. This is what we term *vertical mobility* (or policy mobility, as defined in [15]), where a node can be in an environment of several overlapping networks with different properties and choose freely which network to use. In our definition of vertical mobility we do not require the networks to be heterogeneous, as we would also be interested in the possibility to change between, say, two overlapping WLANs with different properties.

For vertical mobility we define the unit of mobility to be a network connection, which we define as a logical mobile entity that can move between networks. This fits our focus on mobility as an atomic change of location, as identified in the previous section. Subsequently, we view vertical mobility as a type of logical mobility. They both share the characteristic that they can be controlled by the application designer.

3 Requirements for the Modeling Framework

We are interested in providing a framework for modeling mobile information systems. Specifically, we are interested in modeling concepts that are useful when designing applications and that allow us to leverage all the new possibilities that mobility brings. This is also known as *adaptive, mobility-aware* applications.

In our approach, we are interested in an entity's change of location and its ability or willingness to move. Consequently, we can reason about both reactive and proactive adaptation. We do not consider the continuous movement of entities, but only the result of it, i.e., location change. Furthermore, we view location as a defined entity

with boundaries that can contain other entities. Following this definition, we do not consider location by satellite positioning, as in location aware systems, to be a location entity, but rather one of several properties that might describe a location.

The framework should separate between the different types of mobility that have previously been identified; physical and logical mobility. In addition, we believe that a framework for modeling mobile information systems should also include the concept of vertical mobility, as change of network is very relevant to mobility and adaptive applications. Our goal is to propose a user-friendly and visual modeling framework that allows developers to reason and communicate about these types of mobility in mobile systems.

We do not have the opportunity to go into a detailed discussion of requirements here, but for a fine-grained list of requirements and the reasoning behind them, see [1].

4 Proposed solution

4.1 Mobility Metamodel

Grassi et al. [3] define the following issues that need clarification when we want to model mobility:

- Which entities move?
- How do we model the movement of an entity?
- What causes the movement of an entity?

As we defined in the requirements, we view mobility as an entity's willingness and ability to move and the actual location change of these entities. First, we introduce location as a concept. By location we mean any entity that has some concept of a boundary and that can contain other entities. A location can be divided into physical and logical locations. Examples of locations we are interested in separating between are **places**, **networks, devices** and **execution environments** (such as a virtual machine like JVM), as illustrated in Figure 1.

Second, we need to identify the entities with the ability to change location. Mobile entities are also divided into physical and logical elements, and indicative examples of interest are **devices, people, locations** (e.g. vehicles), **network connections** and **software**. See Figure 2. Our metamodel does not include concepts for detailed modeling of the network topology (like routers, proxies, multiplexes, etc), as we, from an adaptive application's point of view, are only interested the different networks the application has access to and their characteristics.

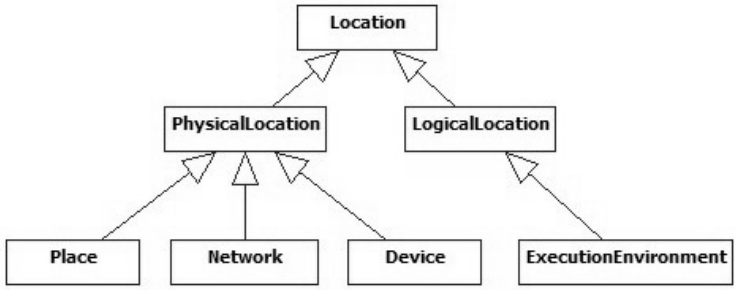

Fig. 1. Location metamodel.

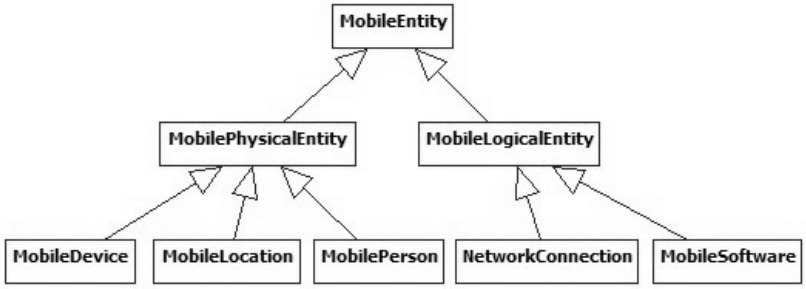

Fig. 2. Mobile entity metamodel.

Third, Figure 3 illustrates how these concepts relate. A location is an entity that can contain other entities. These entities can be stationary or mobile in nature. There is a nesting relationship between locations, where one location can contain several other locations. This relationship can effectively model locations at different levels, like a room contained within a building contained in a city. Most entities will have one location. However, some entities might not have a location, e.g. a top-level location, while other entities might have several locations, e.g. a distributed file system. Mobile entities must have at least one location, and they have the ability to move between locations that are connected. The semantics of being connected varies for the different types of mobility, which is explained in the next section.

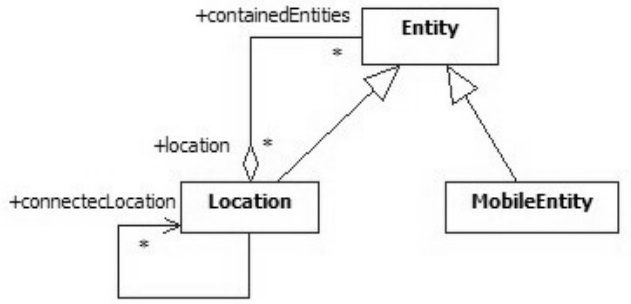

Fig. 3. Mobility metamodel.

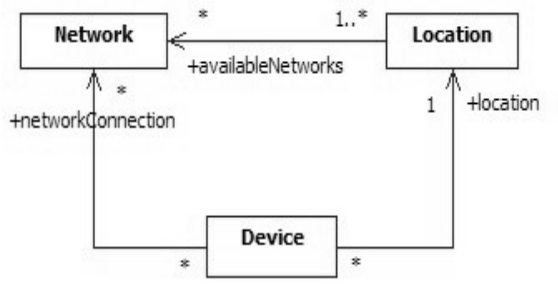

Fig. 4. Vertical mobility metamodel.

Several of the entities in our mobility domain can play different roles depending on the selected viewpoint. A mobile device is considered an entity that can change location from the viewpoint of physical mobility, while it has the role of a location that mobile code potentially can move to and from in the context of logical mobility.

Figure 4 shows a conceptual model for vertical mobility, which is somewhat different from general mobility. A network can not contain another network like locations can contain other locations. In addition, vertical mobility does not only involve the mobile entity (network connection) and its container (network), as the device and its location has to be considered as well. A physical location is associated with the available networks at that location, while a device is associated with the network it is currently using. A device is thus aware of its available networks through its location.

4.2 UML Profile for Modeling Mobility

The profile presented in Figure 5 is inspired by the profile introduced by Grassi et al. in [3], which is a profile for modeling physical and logical mobility. A detailed dis-

cussion of the differences of our approach and that of Grassi et al. is provided in Section 5.2.

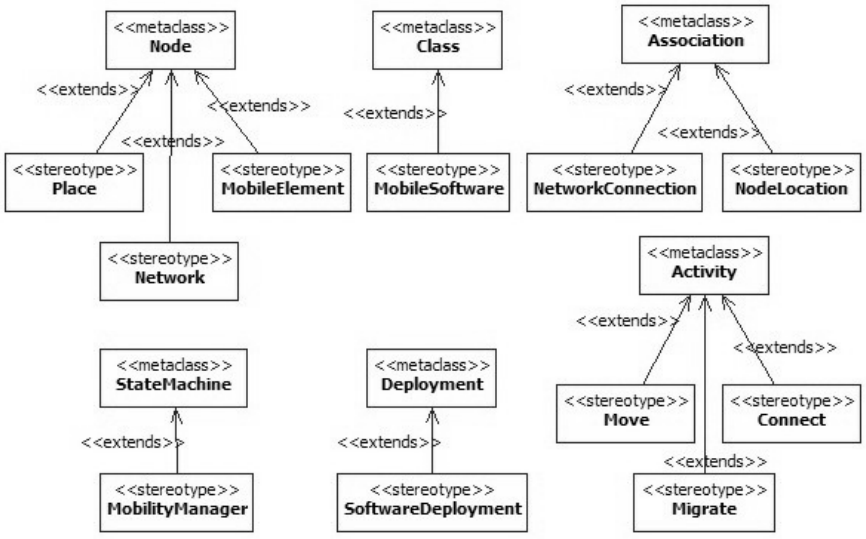

Fig. 5. UML profile stereotypes for mobile systems.

The previous section identified the entities we consider for physical, logical and vertical mobility. For devices, users and other physical mobile entities we use the stereotype MobileElement [3]. Furthermore, we introduce the stereotypes MobileSoftware and NetworkConnection. These three stereotypes extend Node, Class and Association, respectively. Consequently, MobileSoftware can be used on both classes and components to denote a piece of mobile software.

Mobile entities move between locations. The UML2 specification has already defined constructs for the Device and ExecutionEnvironment concepts, which are locations for MobileSoftware. We introduce the stereotypes Place and Network to denote physical locations and overlapping access networks.

Each entity can have a location. NodeLocation is a stereotyped Association that specifies the location of a node. The location of mobile software is modeled through the SoftwareDeployment stereotype.

The movement of mobile entities is modeled by extensions to the Activity metaclass. These are Move, Migrate and Connect for physical, logical and vertical mobility, respectively. A mobile entity can only move if there exists a channel connecting the two locations. This could imply a corridor connecting two rooms for physical mobility, or two nodes being connected to the same network or the Internet for logical mobility. For vertical mobility, both networks have to be available from the device's current location.

As presented in [3], we use the concept of a mobility manager. The MobilityManager stereotype is a state machine for modeling the cause of mobility. The intention is

that a system can change its mobility policies by selecting between different mobility managers. It is worth noting that a mobility manager only covers adaptation through mobility. Our profile does not try to cover adaptation in general.

Table 1. Profile stereotypes.

Stereotype	Extends	Constraints	Description
MobileElement	Node	Can be located in a Place.	Has the ability to be moved between physical locations.
MobileSoftware	Class	Can only be located in a Node.	Has the ability to be moved between nodes.
NetworkConnection	Association	Connects a Device to a Network.	Has the ability to be moved between networks. Can be changed by Connect.
Place	Node		A physical location that can contain other entities.
Network	Node		Networks can span several locations and devices can connect to them through NetworkConnection.
Move	Activity	Locations must be connected.	Moves a MobileElement between two physical locations.
Migrate	Activity	Locations must be connected.	Moves a MobileSoftware between two nodes.
Connect	Activity	Destination Network must be at Device's NodeLocation.	Moves a NetworkConnection between two networks.
NodeLocation	Association	Connects a Node to a Place.	Specifies the location of a MobileElement. Can be changed by Move.
SoftwareDeployment	Deployment	Deploys a Component to a Node.	Specifies the current deployment of a MobileSoftware. Can be changed by Migrate.
MobilityManager	StateMachine		Models the causes and triggers of the movement of mobile entities.

5 Validation

In [1] we validated our profile through a case study. In the following, we present our experiences and lessons learned from the case study, in addition to positioning our profile among related work on the topic.

5.1 Case Study

In [1] we conducted a case study where we used the profile to develop a mobile information system. A PIM was designed before being marked with stereotypes from the UML profile. A part of this design was then transformed from PIM to PSM (platform specific model) and all the way to code. For the PIM to PSM transformation we used the ATLAS Transformation Language (ATL) [16], while we used MOFScript [17] for the PSM to code transformation.

In the case study we designed two deployment diagrams – one with and one without the use of stereotypes from our profile. While the first diagram only models one static scenario, the second diagram represents a snapshot of a possible scenario, while also showing other scenarios that are possible due to physical, logical and vertical mobility.

This type of model can serve two purposes; as a design time and a runtime model. In our case it was used as the former. Applying the profile resulted in a model that describes an important part of the application domain for an adaptive system. The mobility and location of a mobile entity will heavily influence the resources and context available to the system, giving the designers a fuller understanding of the environments the system will run in and needs to adapt to.

An adaptive system can also maintain a runtime version of the model, always keeping track of its current location and context. By analyzing previous mobility patterns or a schedule, the application could also offer adaptation based on future location and context. This area of use has been explored in the FAMOUS project, without seeing realization in the middleware.

We also designed a class diagram of the client application and marked a class as being mobile. Based on the transformation mappings we defined, we transformed the class diagram into a simple mobile code solution for Java Micro Edition (J2ME). The transformations did not result in a running application, but showed how marking a piece of software as mobile at the PIM level can automatically produce application solutions through transformations.

With the use of transformations, development time was naturally significantly shorter than it would have been to manually create all the models and code. In addition, the developer does not need to have any knowledge about the platform. However, developing transformations requires both time and expert knowledge of domains and platforms. As the number of platforms is significant for mobile devices and new devices are introduced at a rapid pace, one must consider the time and resources spent on implementing a MDD approach versus time saved using it.

The last part of the design phase was designing mobility managers for the different types of mobility, which specified the different causes and triggers for the mobility and transitions between the different scenarios modeled in the mobility deployment

diagram. The drawback of using state diagrams is that they model state changes based on simple event-condition statements. Sometimes, decision making about which adaptation strategy to use is a complex calculation. In the MADAM middleware, for example, utility functions might draw information from numerous context sources to determine the best adaptation strategy for a given context [18].

5.2 Related Work

The literature contains several approaches to modeling mobility. In the following, we give a brief overview of some of these and show what our approach contributes with.

In [19] UML sequence diagrams are extended to model complex mobility patterns, but this requires a nonstandard extension of UML sequence diagrams. The diagrams provide the possibility to abstract away from irrelevant details. Their semantics is similar to that of ambients in that a mobile object is a location and a mobile process as well [20].

In [20], UML class and activity diagrams are extended, allowing the representation of mobile objects and locations as well as basic primitives such as moving or cloning.

Most relevant for the approach presented in this paper, though, is Grassi et al.'s UML profile for modeling mobile systems [3]. It makes a clear distinction between logical and physical mobility, and these concepts have their own representations.

The most significant difference between the approaches in [3] and this paper is the introduction of metalevel concepts for vertical mobility. The network a device is connected to has significant effects on the context a system experiences and the adaptation strategy it employs. By allowing developers to reason about different, overlapping networks in their models, we believe they will have a better vocabulary for reasoning about mobility and adaptivity in mobile systems.

When it comes to modeling physical and logical mobility, the approaches are similar except for a few differences.

Grassi et al. use the stereotypes MobileElement and its inherited stereotype MobileCode to model physical and logical mobile elements, respectively. They neglect to extend any metamodel classes for these concepts. We remedy this situation in our profile. In addition, we deemed the inheritance relationship as unnecessary and removed it, and renamed MobileCode to MobileSoftware as we think the latter puts less restrictions on the use of the concept.

Place, NodeLocation and MobilityManager are the same in both profiles. CurrentDeployment has been renamed SoftwareDeployment to better reflect the naming convention used for NodeLocation. In [3] the concept MoveActivity is used for moving MobileElements, while this is further specialized into PhysicalMove and LogicalMove in [21]. We used the terms Move and Migrate for the same meaning.

In [3], the authors introduced the stereotyped deployment AllowedDeployment, which is used to model additional constraints, like security and administrative domains, to the mobility of mobile code. We do not, however, see any reason for treating logical mobility any differently from physical mobility in this respect. As security is outside our scope, we chose not to include AllowedDeployment or any similar constructs.

Grassi et al. also specifies a set of Activity stereotypes that supports more fine-grained concepts and operations related to mobility and management of a mobility model; BeforeMoveActivity, AfterMoveActivity, AbortMoveActivity, AllowDeploymentActivity and DenyDeploymentActivity. We have not treated these in this paper.

The following table lists the stereotypes presented in this paper and the corresponding stereotypes in Grassi et al.'s profile.

Table 2. Comparison to earlier work.

UML profile for mobile systems	Corresponding concepts in Grassi et al.'s profile
MobileElement	MobileElement
MobileSoftware	MobileCode
NetworkConnection	None
Place	Place
Network	None
Move	MoveActivity/PhysicalMove
Migrate	MoveActivity/LogicalMove
Connect	None
NodeLocation	NodeLocation
SoftwareDeployment	CurrentDeployment
MobilityManager	MobilityManager

6 Conclusions and Future Work

Mobile computing is characterized by a high level of heterogeneity and significant variations in available resources. As a result of this, it is generally accepted that mobile systems should be able to adapt to changes in context and resources.

Based upon earlier work, we presented a UML profile for modeling mobile information systems. The focus has been on modeling mobility as a change of location, and how a mobile system can adapt to its changing environment. The profile differentiates between and provides concepts for physical, logical and, as included in this paper, vertical mobility. Our approach is based on deployment diagrams, where we model the relationships between locations and mobile entities. Mobility managers, as defined in [3], are state machines that drive the mobility of a system. Based on events

like location change, change in battery levels or network quality, the mobility managers can decide to employ a mobile code strategy or connect to another network.

In [1] we used our framework to develop a case study application. This provided us with valuable information about the usefulness of the framework and was a basis for its validation. In this paper we discussed the experiences we gained from the case study, before giving an overview over related work on the topic. The major contribution from our profile is the introduction of vertical mobility. To further validate the proposed framework we should perform additional case studies to assess its usefulness in different kinds of and more complex systems.

References

1. Dehlen, V. *Developing Mobile Information Systems*. 2006, University of Oslo: Oslo. p. 145.
2. Chalmers, D. and M. Sloman. *A Survey of Quality of Service in Mobile Computing Environments.* IEEE Communications Surveys, 1999.
3. Grassi, V., R. Mirandola, and A. Sabetta. *A UML Profile to Model Mobile Systems*, in *2004 - The Unified Modelling Language*. 2004, SpringeLink. p. 128-142.
4. Satyanarayanan, M. *Pervasive Computing: Vision and Challenges.* IEEE Personal Communications, 2001.
5. Patterson, C.A., R.R. Muntz, and C.M. Pancake. *Challenges in Location-Aware Computing.* IEEE Pervasive Computing 2003. **2**(2): p. 80-89.
6. Roman, G.-C., G.P. Picco, and A.L. Murphy. *Software engineering for mobility: a roadmap*. in The Future of Software Engineering. 2000. Limerick, Ireland.
7. Kristoffersen, S. and F. Ljungberg. *Mobile Informatics Innovation of IT Use in Mobile Settings: IRIS'21 Workshop Report.* SIGCHI Bulletin, 1999. **31**(1).
8. Küpper, A. and O. Spaniol. *Evaluation of strategies for supporting personal mobility and service portability*. in 2000 IEEE Service Portability and Virtual Customer Environments. 2000.
9. Sun, J.-Z. and J. Sauvola. *On fundamental concepts of mobility for mobile communications.* in 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. 2002. Lisbon, Portugal.
10. Cardelli, L. and A.D. Gordon. *Mobile Ambients*. in First International Conference on Foundations of Software Science and Computation Structure. 1998.
11. Cardelli, L. *Abstractions for Mobile Computation*. 1998, Microsoft Research, Microsoft Corporation.
12. Ylianttila, M. *Vertical handoff and mobility - system architecture and transition analysis*. 2005, University of Oulu: Finland. p. 70.
13. Stemm, M. and R.H. Katz. *Vertical handoffs in wireless overlay networks.* Mobile Networks and Applications, 1998. **3**(4).
14. Bellavista, P., M. Cinque, D. Cotroneo, and L. Foschini. *Integrated support for handoff management and context awareness in heterogeneous wireless*

networks. in 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing MPAC '05. 2005: ACM Press.

15. Tourrilhes, J. *L7-mobility: a framework for handling mobility at the application level*. in 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. 2004.
16. *ATLAS Transformation Language (ATL) homepage*. http://www.eclipse.org/gmt/atl/
17. *MOFScript homepage*. http://www.eclipse.org/gmt/mofscript/
18. Paspallis, N. and G.A. Papadopoulos. *Distributed Adaptation Reasoning for a Mobility and Adaptation Enabling Middleware*. in 30th Annual International Computer Software and Applications Conference (COMPSAC 2006). 2006: IEEE Computer Society
19. Kosiuczenko, P. *Sequence diagrams for mobility*. in ER/IFIP 8.1 Workshop on Conceptual Modelling Approaches to Mobile Information Systems Development (MobIMod). 2002. Tampere, Finland: Springer.
20. Baumeister, H., N. Koch, P. Kosiuczenko, and M. Wirsing. *Extending Activity Diagrams to Model Mobile Systems*. in Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World. 2002: Springer-Verlag.
21. Grassi, V., R. Mirandola, and A. Sabetta. *UML based Modeling and Performance Analysis of Mobile Systems*. in 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems. 2004: ACM Press.