Model-based Ubiquitous Interaction Concepts and Contexts in Public Systems

Thomas Schlegel, Christine Keller

Abstract. Ubiquitous systems and interaction concepts are increasingly finding their way into public systems like shopping malls, airports, public transport or information kiosks. At the same time, these user interfaces also undergo significant changes. Technologies like multi-touch systems or voice-based interaction are now available to the general public and widely used. In ubiquitous systems, these modalities are often combined, sometimes even dynamically at runtime. This leads to new challenges for the conceptualization and development of ubiquitous user interfaces in public systems, especially where this implies adaptive behavior. We present contexts that possibly influence the interaction with such public systems and describe ways of modeling this interaction integrating context-adaptivity already in the interaction models of public systems. Taking into account the context of the public system and its users, we extend the concept of Interaction-Cases to contain model aspects for different interaction contexts in public systems.

1 Introduction

Public systems represent a field of application for ubiquitous techniques that is growing recently. Ubiquitous environments are integrated into public information technology and socio-technical public systems like shopping malls, airports or public transport [1,2]. Some of them have particular target groups, like tourists or handicapped people [3,4]. Others are designed to serve the general public [5]. New interaction technologies like multi-touch interfaces or gesture recognition facilitate the construction of really pervasive computing environments for public systems.

In public places, computing environments must be as unobtrusive as possible. At the same time, their usability and accessibility must be very high and designed to support a variety of strongly different users without training. Combining multiple interaction techniques in ubiquitous environments for public systems aims at supporting both, pervasive and easily accessible computing that is optimized to the situation of the user without the availability of concepts like personalization, profiles etc. found in "classical" interactive systems like business software, home automation or control environments. Ubiquitous systems offer new opportunities for public settings, which will make them find their way into public systems, but also bring new challenges with them, regarding their development and modeling.

Ubiquitous systems support many different interfaces and often need to be context-aware to optimize their interactive behavior regarding context. They are characterized by integrating several means of interaction and being highly adaptive. Here, a model-based approach can lead to efficient development of ubiquitous systems and support adaptive design of user interfaces in ubiquitous systems [6]. However, model-based

design methods like Model-Driven Architecture (MDA) are often not flexible enough to meet the needs of designing adaptive, context-aware ubiquitous systems [7]. They also focus on design-time generative approaches, which are not applicable to non-monolithic, modular systems with changes on runtime. Therefore, a modeling-technique for interactive public systems is needed, which supports context-aware and highly adaptive interactive ubiquitous systems, in order to facilitate the creation of ubiquitous public systems that combine context-adaption as well as multiple and multimodal interaction techniques.

In traditional software engineering, user and interaction models are specified at design time, using persona, textual descriptions or just the mental models that user interface designers and developers have about the future users of the system [8,9]. Based on these user models and the identified roles and actors, the whole system is then designed [10,11]. In ubiquitous public systems, however, the users can often not be associated with a specific user group. In public, all kinds of people are around and can turn into users of the public system, often with different or unspecific needs not covered with a dedicated task or system, making the user anonymous and unpredictable at design-time. However, the context of the public system and its user interface is known, can be specified and observed at runtime. Based on the context the public system observes, such as location, time or input, the system can adapt, for example, its interaction modalities. Therefore, especially in public systems, the modeling of an interactive and ubiquitous system has to focus on system context.

The field of context-aware systems is very broad, being researched for several years now. Especially the growth of interest in ubiquitous systems has pushed the development of context-aware environments [12,13]. One of the first context-aware systems was the *active badge location system*, developed by Want et al. in 1992 [14]. It observed the user's location and redirected telephone calls concerning this location. The early context-aware systems mainly considered location as the context of the user, like many tourist guides, for example [3,15]. Later on, other contexts were considered as well, which led to different approaches on modeling context. Dey et al. for example, identified several popular categories of context such as "location, identity, activity and time" [13]. The classification of context that is necessary for building context-aware systems led to the development of several ontologies for context, for example by Chen et al. and also by Moore et al. [16,17]. Many context-aware ubiquitous systems emerged and this development led to the design of several frameworks that facilitate the construction of context-aware systems [18,19].

The specifics of public systems were considered in some ubiquitous systems that were designed for public environments, for example, systems that support handicapped people in settings like public transport [4]. Other systems, like the GUIDE project, focused on tourists, or on students on a campus, like the e-campus project [2,3]. Some other projects involving public transport settings used ubiquitous and context-aware technologies [15,20]. However, to our knowledge there is little research about the properties and contexts of public systems that does not focus on specific settings or user groups, like tourists, students, public transport etc. We therefore define a general public system as a system that performs in public spaces and does not target specific user groups but is available to all people, i.e. the public.

Concerning modeling techniques for interactive systems, there is extensive work on MDA and model-driven interface design [6,7]. As noted above, some extensions of

UML have led to modeling languages for web applications or web services, some of them context-aware [21,22]. Most of this work, however, does not consider context-awareness and adaptivity in ubiquitous public systems.

In this paper we therefore present an overview on possible contexts in public systems and a basic taxonomy of these contexts. We then introduce a model-based approach on designing context-adaptive interaction in ubiquitous public systems.

3 Dimensions of Context in Public Systems

Since in public systems, personalization often is impossible, they strongly depend on context in order to adapt to user's needs and the surroundings. Models of ubiquitous public systems should be adaptive regarding these different contexts. As a basis for adaptive modeling techniques, the possible contexts of public systems need to be analyzed. We developed a classification of contexts of public systems for this purpose, which is shown in figure 1. The context classification is still work in progress and should not be too fine-grained for serving as a basis for different systems and context ontologies. It shows our basic approach to modeling context and points out the specific properties of public systems that must be taken into account while modeling interactive ubiquitous systems for public settings. Ideally, context models are defined in a domain-specific way for the specific public system, but rely on a common basic ontology that allows for matching and integrating the context ontologies on the top level of abstraction. The examples of context-adaptive interaction models for ubiquitous public systems, which we describe in section 5, show that already a coarse-grained and incomplete context classification can be applied successfully to our modeling technique.

Most ubiquitous systems have a kind of **Interaction Context** (**I**), consisting of the system's *Input Context* (*In*), representing the possibilities of input, i.e. of entering, selecting or editing information. The system also has a *Processing Context* (*Proc*) that reflects the processing of input on part of the system, including capability models and sensor fusion. Analogously, there also exists an *Output Context* (*Out*), which relates to different output modalities. Especially in public areas, the context of interaction by the user has to be considered as well, in particular as there are aspects that are known to the system without specific knowledge about the user. A user not only perceives and acts while interacting with a public system; he also processes what he has perceived. This leads to the *Perceptive*, *Cognitive* and *Acting Context* (*Per, Cog, Act*) of the user as important contextual aspects. In public, the user can be distracted by noise, too much light (e.g. glare) as well as bad lighting, for example. These conditions influence his *Perceptive* as well as his *Cognitive Context*, like being in a hurry or looking for somebody else, which will reduce or divert his attention and therefore will reduce his available cognitive and perceptive capacity.

Of course, the **Spatial Context** (**Sp**) must be considered, too. This concerns the location of the user as well as the location of the system. Large public displays as well as the mobile devices of users may be a component of a public system. The location of the system therefore can be *fixed (Fix)* or *moving (Mov)*, but can also influence size and visibility of an available display. We furthermore identified the **Temporal**

Context (**Temp**) with its sub-contexts *Absolute Time* (*Abs*) and *Relative Time* (*Rel*). Relative time will occur, for example, where a distance separates the user from a destination, leading to a relative time needed to get there depending on transportation, or where interaction occurs relative to an event like the late arrival of a train.

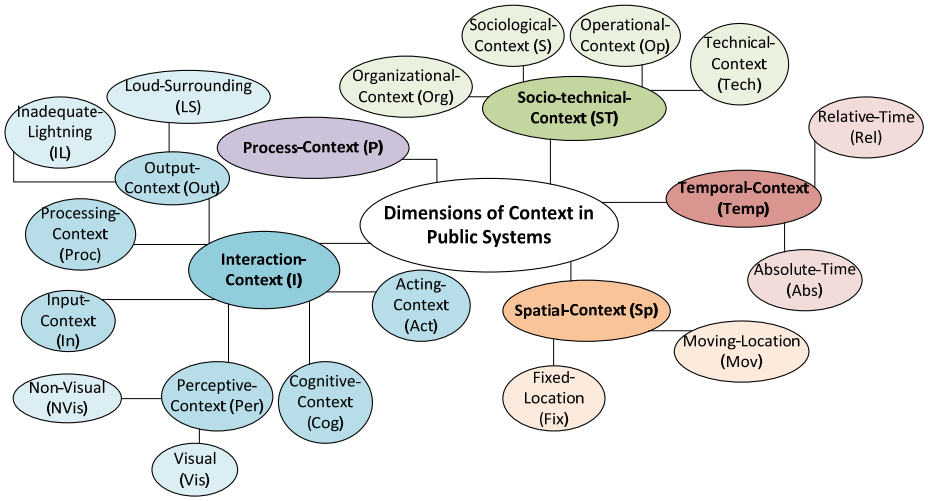

Fig. 1. Dimensions of Context in Public Systems

A context specific to and of high interest in public systems is the **Socio-Technical Context** (**ST**). In public systems, there is a *Sociological Context* (*S*), considering the common use of resources by many people. There are social rules in public spaces that are followed by most of the people there. These rules can affect the usage of public ubiquitous systems. Concerning, for example, the interaction using a big public display, the aspect of privacy has to be considered. A user should not be required to enter or request personal data in such a way that might disclose it to others [23].

The *Technical Context* (*Tech*) is also a sub-context of the Socio-Technical Context and describes the technical abilities of the system. In public transport this could be the question whether a subsystem has fast and reliable access to an up-to-date timetable data source, including delays or failures, or only to the regular timetable.

Another sub-context of the Socio-Technical Context of public systems is the *Operational Context* (*Op*) of a system. This covers everything that goes on "behind the scenes" and includes, in public transport for example, activities in the railway control center. This context affects public systems, because many of them depend on certain operational procedures and exhibit only a small part of the full socio-technical system to the public.

The *Organizational Context* (*Org*) models the conditions and contexts of organizations involved. In public spaces, these could be the operating company of public transport as well as organizations running shops at airports or railway stations. For services, the Organizational and the whole **Process Context (P)** are of high importance. Process Context in public systems is defined upon user and system actions and their structural and logical interdependencies (causality etc.). Many actions exhibit dependencies or will trigger other actions necessary to accomplish a

complete task with the desired result. Especially regarding service-oriented architectures, the Process Context is of high value for service quality and many other aspects like security and safety.

This also shows that many of the context types, like Organizational Context and Technical Context are interconnected or may even be integrated to form, in this example, the Socio-Technical Context. Therefore, one must consider the modeling of contexts as well as their integration or even fusion.

4 Interaction-Cases

When software systems with interactive components are designed, often stakeholders from different disciplines participate in the process. The design step therefore requires techniques to be easily understandable as well as highly applicable in different phases and aspects from informal requirements to formal processes, in order to be used by technical as well as non-technical stakeholders. Paper-based prototypes, scenarios and textual descriptions of a system's behavior are non-technical and easily understandable approaches for early design phases [10,24]. However, these artifacts lack formalization and therefore cannot be linked to artifacts of later development stages like code. Often the early stage design artifacts become outdated and inconsistent due to changes in later artifacts, which cannot be tracked back to the descriptions in early specification artifacts.

The Unified Modeling Language (UML) provides a semi-formal way of modeling and specifying software in several development stages, from informal Use Case definitions to class diagrams and generated code stubs. Standard UML Use Cases provide a non-technical means of specifying the behavior of a software system. Since they are based on textual descriptions, they are easy to understand also for non-technical participants of the design process. The Use Cases can then be used to create more formal specifications, like class diagrams and could serve as a link to a formalized system specification.

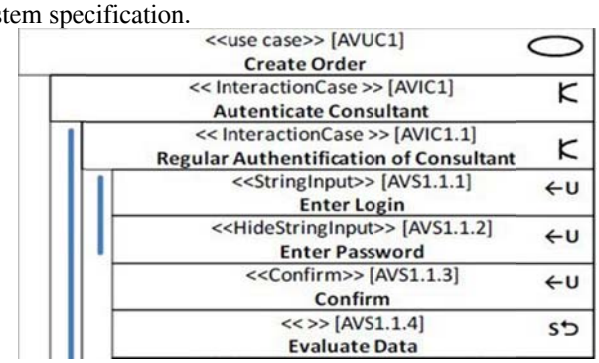

Fig. 2. Part of the specification of an authentication process: Interaction-Case "Authenticate Consultant" within the Use Case "Create Order"

Interactive components, however, are still characterized as plain textual descriptions within Use Cases. In earlier work we therefore proposed **Interaction-**

Cases as a technique based on Use Cases, designed to support an easily applicable and at the same time formally strong means of specification of interaction procedures [25]. Interaction-Cases were created to support incremental and iterative development and to enable developers and designers to start modeling on paper, being able to formalize and refine the Interaction-Cases in later stages of development and link them to other artifacts of the development process.

An Interaction-Case consists of sequences of Interaction-Cases and/or (atomic) Interaction-Steps. By nesting Interaction-Cases, a sequence of interactions can be modeled coarse-grained in early design stages and can then be refined into more detailed Interaction-Cases or, vice versa, a sequence of atomic Interaction-Steps can be aggregated later on. Interaction-Cases and Interaction-Steps have an identity and can therefore be referenced and reused by other Interaction-Cases using this identifier. Interaction-Case components can inherit from other components, thus introducing object-oriented type semantics and enabling powerful re-use. The parent class of an Interaction-Case can be noted like an UML stereotype (<<type>>), as shown in figure 2.

Applying the object oriented paradigm further, abstract Meta Components are introduced, identified by a "?" in front of their name. The concept of Meta Components is similar to the concept of interfaces or abstract classes in object oriented programming: A Meta Component specifies certain characteristics of an Interaction-Case, but must later be realized by inheritance to form a concrete one.

The steps of an Interaction Case can be modeled to be executed sequentially, in parallel or without a predefinition of the execution order at design time.

It is possible to direct the flow of Interaction-Cases in branches. There are three possible types of branches. Conditional branching initiated by the user (?DECIDE), initiated by the system (?ConditionalBranch) and unconditional branching by the system (?GOTO). Conditions for conditional branches are followed by a question mark. In parallel execution it is possible to execute all branches, m out of n and to model options for the user to decide (at least one, not all or one out of n).

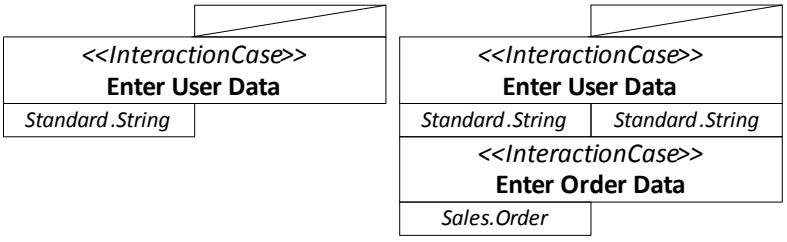

Fig. 3. Interaction-Cases with Input- and Output-Container

In addition to the concepts described in [25], we extended this basic definition of Interaction-Cases by adding an Input and a n Output Container describing the data type of the component's input and output (ObjectType). The Containers are visualized by boxes on top of and below an Interaction-Case, as shown in figure 3. The top box describes the input consumed by a component, e.g. the data that is presented to the user in the interactive step, while the box on the bottom of the Interaction-Case or Interaction-Step describes its output, e.g. the information gathered from the user or read from a sensor or database. The ObjectTypes of input and output of a component

are inherited from its parent class. If no input or output is defined, the respective box is crossed out (figure 3). The drawing also shows that the input ObjectType of a successor has to match the output ObjectType of its predecessor. The visualization of input and output-Containers easily ensures consistency of information objects even in early sketching phases.

Also, an extension to the Microsoft Visual Studio 2010 Editor for UML Use Case diagrams was developed, which facilitates the integrated modeling of Interaction-Cases and Interaction-Steps within the development environment. Figure 4 shows a screenshot of the editor extension that was developed.

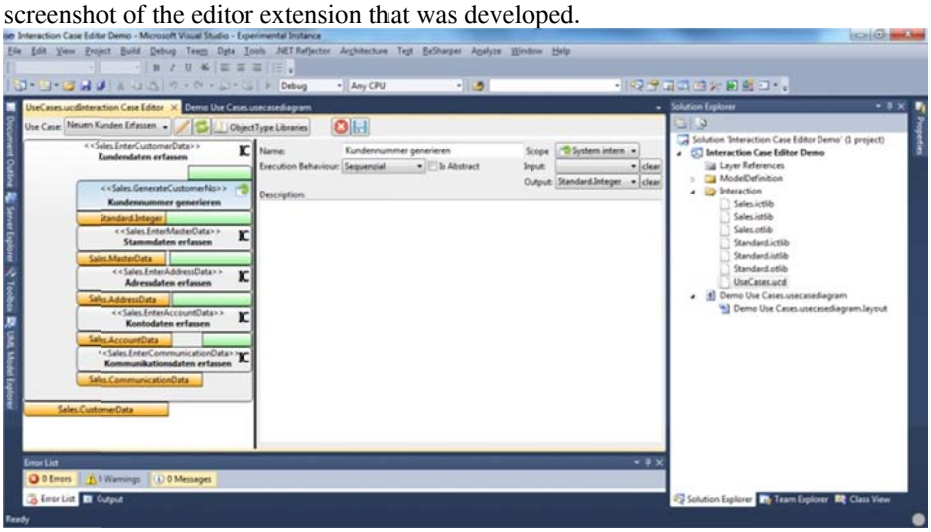

Fig. 4. Screenshot of the extension of the Visual Studio Use Case Editor

Types of Interaction-Steps or Interaction-Cases are defined by inheritance from a parent Interaction-Step or Interaction-Case. Type definitions for Interaction-Step types and Interaction-Case types can be stored in libraries and imported into other projects as well, to facilitate reuse. Based on the present possibilities for modeling Interaction-Case types, Interaction-Step types and ObjectTypes, the generation of code stubs becomes possible, which would further increase the integration of Interaction-Cases into the development of interactive systems.

5. Context-adaptive Interaction-Cases

In order to support ubiquitous public systems with context-based modeling, we extended the Interaction-Case modeling technique so that Interaction-Cases can be modified by certain Context Types. In early development stages it is possible to denote Interaction-Cases without defining all Interaction-Steps, which supports incremental modeling. Already in this phase, Interaction-Cases can be labeled with Context Modifiers, to indicate that the refinement of this Interaction-Case depends on certain types of context. In the visual representation of an Interaction-Case, a Context Modifier is indicated by the abbreviation of the Context Type, written in a semi-circle

shape that is placed on the left side of the Interaction-Case drawing, as shown in figure 5 on the left.

Alternative Interaction-Cases can now be modeled for different instantiations of the denoted Context Type. This allows to model different context-dependent forms of system behavior. In order to allow Interaction-Cases to be substituted by their context-modified equivalent, these equivalents should be specializations of a common Interaction-Case super-type. Of an Interaction-Case that is modified by the Perception Context (**Per**), alternative Interaction-Cases can be modeled, for example for the Context Types Visual (**Vis**) and Non-Visual (**NVis**). This allows, for instance the modeling of a ubiquitous public system that switches to non-visual (e.g. speech-based) interaction, in case the Perception Context of the user is Non-Visual. This can occur if the user is blind, visually impaired or maybe running towards a train, looking for the right number.

Context-adaptive systems should of course not only adapt to one type of context at a time. Therefore, it is possible to model Interaction-Cases that have multiple Context Modifiers. In that case, the abbreviations of the different contexts that influence the interaction component are noted within the semi-circle shape. For different combinations of Context Types, different modified Interaction-Cases can be modeled. However, since many different combinations of Context Types are possible if there is more than one Context Modifier, not all combinations must be modeled as separate Interaction-Cases. The system or the developer can choose the most specialized context-modified Interaction-Case available that fits the context currently observed.

An example would be an Interaction-Case that has a Context Modifier based on Perception Context (**Per**) and also a Context Modifier based on Output Context (**Out**). It is now possible to model several kinds of Interaction-Cases. One Interaction-Case could be specified for Non-Visual (**NVis**) as sub-context of Perception Context and Loud-Surrounding (**LS**) as sub-context of Output Context. If a blind or visually impaired person would have to use a public system in a loud surrounding, this system then could adapt by not only choosing speech-based interaction, but also by increasing the volume of its speech output as well as the sensitivity and noise reduction of the microphone. Another kind of Interaction-Case could be modeled for Visual (**Vis**) as sub-context of Perception Context and for Inadequate-Lighting (**IL**) as sub-context for Output Context. Instead of switching to speech-based interaction, this Interaction-Case could be realized using enhanced screen brightness.

Exploring the combination of sub-contexts further led us to introduce rule-based sub-contexts. These contexts refine Context Types like Inadequate Lighting or Loud Surroundings, by introducing the parameters that concern these Context Types. It is then possible to define a sub-context of Inadequate Lightning that states “brightness < threshold”. Using such a rule-based sub-context, the abovementioned scenario can be modeled more precisely. The second Interaction-Case can be specialized for “brightness >= threshold”, when enhancing the screen brightness is still effective. An additional Interaction-Case using “brightness < threshold” could then switch the interaction to speech-based.

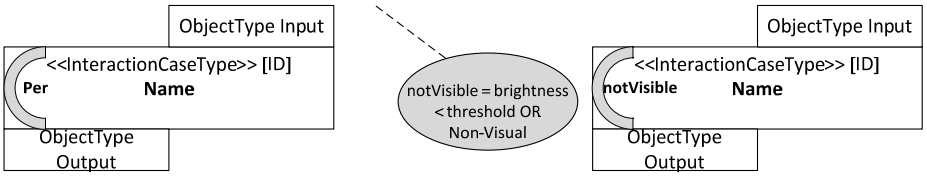

Fig. 5. Context-adaptive Interaction-Case and usage of rule-based sub-context

The combination of contexts so far included combinations using AND logic of multiple inheritance. With the use of rule-based contexts, context combinations can be extended to rule-based sub-contexts like "context1 OR context2", using logical operators different from AND. The logical operators permitted here are OR/NOT/XOR, further logical operators would complicate the modeling too much. In our modeling technique, rules like that are modeled as separate contexts and also used as separate contexts, too. However, they still are modeled as specializations of the basic context types seamlessly integrating them in the taxonomy of context available.

This way, the aforementioned example Interaction-Cases can be expanded, using the rule-based context notVisible = "((brightness < threshold) OR Non-Visual)". Both context conditions then lead to speech-based interaction. This example is also shown in figure 5.

5 Discussion and Future Work

In this paper we have presented context-adaptive Interaction-Cases as a modeling technique that supports the modeling of interactive context-aware and adaptive systems. Using our classification of context in public systems, this modeling technique can be used for supporting the development of ubiquitous public systems, as they emerge in many different public areas, like airports or shopping malls. The modeling technique of Interaction-Cases can already be used in the very early stages of system design and can be applied only using pen and paper. Its design, however, supports the transfer of paper-based models to more formal descriptions. In order to demonstrate this, an extension for the Microsoft Visual Studio 2010 Editor for UML Use Case diagrams, that enables the modeling of Interaction-Cases and Use-Cases in the development environment, has been developed. The modeling technique also supports the iterative development of the interaction model by providing inheritance and specialization of types, incorporating the powerful object oriented paradigm.

We are planning on refining our context classification of public systems in order to better support the development of ubiquitous and context-aware public systems. Our Microsoft Visual Studio 2010 Editor extension is planned to be extended to include our context-adaptive enhancement of Interaction-Cases. We are also planning on studying earlier development phases and to integrate even less formal conceptualization and prototyping techniques into a model-based design process that allows the easy but also powerful construction of ubiquitous interactive systems.

Acknowledgement: Part of this work has been executed under the project IP-KOM-ÖV funded by the German Federal Ministry of Economics and Technology (BMWi). We wish to thank Tobias Grass for the contributions made by his thesis and the colleagues from the Institute for Visualization and interactive Systems (VIS) .

References

1. Peterson, M.: Pervasive and Ubiquitous Public Map Displays. (2004)
2. Storz, O., Friday, A., Davies, N., Finney, J., Sas, C., Sheridan, J.: Public Ubiquitous Computing Systems: Lessons from the e-Campus Display Deployments. IEEE Pervasive Computing 5(3), 40-47 (2006)
3. Cheverst, K., Davies, N., Mitchell, K., Friday, A.: Experiences of developing and deploying a context-aware tourist guide: the GUIDE project., pp.20-31 (2000)
4. Klante, P., Krösche, J., Boll, S.: AccesSights – A Multimodal Location-Aware Mobile Tourist Information System. In Miesenberger, K., Klaus, J., Zagler, W., Burger, D., eds. : Computers Helping People with Special Needs 3118. Springer (2004) 627-627
5. Bertolotto, M., P., G., Strahan, R., Brophy, A., Martin, A., McLoughlin, E.: Bus Catcher: a Context Sensitive Prototype System for Public Transportation Users., pp.64-72 (2002)
6. Pérez-Medina, J.-L., Dupuy-Chessa, S., Front, A.: A Survey of Model Driven Engineering Tools for User Interface Design. In Winckler, M., Johnson, H., Palanque, P., eds. : Task Models and Diagrams for User Interface Design 4849. Springer (2007) 84-97
7. Kruchten, P., Obbink, H., Stafford, J.: The Past, Present, and Future for Software Architecture. IEEE Software 23(2), 22-30 (2006)
8. Pruitt, J., Grudin, J.: Personas: practice and theory., pp.1-15 (2003)
9. Wasserman, A.: User Software Engineering and the design of interactive systems., pp.387-393 (1981)
10. Aoyama, M.: Persona-Scenario-Goal Methodology for User-Centered Requirements Engineering. Requirements Engineering, IEEE International Conference on 0, 185-194 (2007)
11. Kazman, R., Abowd, G., Bass, L., Clements, P.: Scenario-based analysis of software architecture. IEEE Software 13(6), 47-55 (1996)
12. Prekop, P., Burnett, M.: Activities, context and ubiquitous computing. Computer Communications 26(11), 1168-1176 (2003)
13. Dey, A., Abowd, G.: Towards a better understanding of context and context-awareness. (2000)
14. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Transactions on Information Systems 10(1), 91-102 (1992)
15. Hristova, N.: Ad-Me: A ContextSensitive Advertising System., pp.10-12 (2001)
16. Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. Knowl. Eng. Rev. 18(3), 197-207 (2003)
17. Wang, X., Zhang, D., Gu, T., Pung, H.: Ontology Based Context Modeling and Reasoning using OWL., pp.18-- (2004)
18. Salber, D., Dey, A., Abowd, G.: The context toolkit: aiding the development of context-enabled applications., pp.434-441 (1999)

19. Biegel, G., Cahill, V.: A Framework for Developing Mobile, Context-aware Applications., pp.361-- (2004)
20. Banâtre, M., Couderc, P., Pauty, J., Becus, M.: Ubibus: Ubiquitous Computing to Help Blind People in Public Transport. In Brewster, S., Dunlop, M., eds. : Mobile Human-Computer Interaction – MobileHCI 2004 3160. Springer (2004) 535-537
21. Ceri, S., Daniel, F., Matera, M., Facca, F.: Model-driven development of context-aware Web applications. ACM Trans. Internet Technol. 7 (2007)
22. Sheng, Q., Benatallah, B.: ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development., pp.206-212 (2005)
23. Pernack, R.: Public space and transport : a sociotheoretical approach. Wissenschaftszentrum Berlin für Sozialforschung gGmbH (2005)
24. Barbosa, S. D. J., Paula, M. G.: Interaction Modelling as a Binding Thread in the Software Development Process. (2003)
25. Schlegel, T., Raschke, M.: Interaction-Cases: Model-Based Description of Complex Interactions in Use Cases. (2010)