Towards Requirements Engineering for Context Adaptive Systems

Wassiou Sitou, Bernd Spanfelner
Technische Universität München, Department of Informatics,
Boltzmannstr.3, 85748 Garching/Munich, Germany
{sitou|spanfeln}@in.tum.de

Abstract

Building realistic end user scenarios for ubiquitous computing applications entails large up-front investments. Many context adaptive applications so far fail to live up to their expectations. Firstly, this is due to poorly conceived development tools and methods compared to other, more mature domains. And secondly, they seem to be particularly prone to problems related to a discrepancy between user expectation and systems behavior. This unwanted behavior prevents the vision of an emerging trend of context aware and adaptive applications in ubiquitous computing to become reality. A good understanding of business and customer's requirements may be of immense importance. This paper presents a model-based requirements engineering approach to systematically analyze and specify the basic system behavior as well as the adaptation behavior starting from customer and business needs.

1. Introduction

The increasing technological progress and the associated integration of software systems in a wider range into our everyday life require for more flexibility and multi-functionality of the systems. We expect in the near future that the systems primarily adapt to the needs and wishes of the users. These expectations are often described as important challenges for the emerging field of ubiquitous computing [34, 10, 27, 13]. Ubiquitous computing so far has many different definitions. In fact they are all based on a substantially more flexible system understanding where the needs and wishes of the user are in the foreground. Ubiquity in this sense means enhancing usability of a functionality in as many situations as possible e.g. to overcome certain restrictions which depend on the current situation like limited interaction capabilities or technical resource constraints. The concept of adaptation is used as enabling technology to archive this goal. We denote *context adaptive systems - CAS* as computer-based systems that are capable of recognizing changes in the domain they share an interface with, and at the same time being able to change their behavior to adapt to the changing conditions without necessary direct user interaction. The domain is characterized in terms of perceivable information that is relevant to the adaptation. This characterization is a model of the systems environment and commonly called context.

Many CAS so far fail to live up to their expectations. While performing well in controllable laboratory environments, they seem to be particularly prone to problems related to a discrepancy between user expectation and systems behavior when released into the wild [12]. A reason for this lack is the fact that development methods and tools for such systems are still in an early stage [7]. The characteristics of CAS (context-awareness, pro activeness, changing operational context, changing participants, varying activities etc...) raise the need of defining new methods to suitably elicit, analyze and specify requirements for this kind of systems. In this work, we propose a methodological approach to requirements engineering (RE) for CAS.

In the remainder of this paper, we discuss some related work with a focus on RE approaches for context and adaptivity (section 2). A brief discussion of the conceptual architecture for CAS with its separation of basic functionalities from adaptation behavior is presented in section 3. Then we describe our approach to RE for CAS and discuss its benefits for the development of CAS (section 4). Furthermore we present as proof of concept a long term case study. The paper ends with conclusion in section 6.

2. Related work

Requirements engineering (RE) is considered as an iterative, systematical and interdisciplinary process, coordinated with all involved stakeholders, aiming at providing specifications that satisfy the goals of the majority of users [21]. Although the concept of establishing subsets of requirements matched to different stakeholder groups has been advocated in the viewpoint tradition of RE [19, 25,], the concept of analyzing requirements for individual users and context awareness has not been explored.

With the emerging field of ubiquitous and context aware computing, requirements may not only vary by users, but also by the performed activities and the operational environment. E.g. in location-aware systems, requirements may often change over space and time [5, 1]. The impact of location on requirements was initially explored in the Inquiry Cycle approach [22], where the acceptance of the system's output is stated to be influenced by the location. Change over time is not explicitly modeled apart from concerns over requirements creep and evolution. Of course the importance of goal-oriented methods in RE for CAS should not be neglected. In [18] three RE methods that belong to Goal-Oriented RE (GORE) [33], namely KAOS [6], GBRAM [3] and I* [36], were compared to each other and their applicability in the field of ubiquitous computing were assessed. As expected, it results from this assessment that GORE approaches are certainly interesting for CAS, but definitively not adequate enough to capture, analyze and specify context aware applications. A more advanced step toward a systematic treatment of deferred requirements [15] and also of contextual influences on requirements were introduced by the Clinical RE [14] and the Personal and Contextual RE methods [29]. In software product line engineering a similar problem arises while building systems configurations. In contrast to that area, where only "design time adaptability" is researched, in requirements scoping for CAS the main focus is on the "run time adaptability". This kind of adaptability also includes policy based design for runtime adaptability.

As often stated in the community of context aware computing, context is not only a matter of location. Schmidt et al. stated more explicitly that there is more to context than location [28]. An important aspect of context is the cultural aspect. Although it is recognized that the cultural effects on products are important [20], understanding cultural impact on requirements is still in its infancy. Ethnographic studies suggest that very different requirements arise in situ. Privacy requirements for automatic teller machines for instance are very different between eastern and western societies [8]. Aspects of context regarding user and user groups, their tasks and goals, and the operational environment should not remain unstudied during the RE process for context adaptive systems. Since all these aspects of context and their change over time are not systematically investigated, context adaptive systems will probably often fall into the trap of unwanted behavior [12]. These and other contextual aspects have to be fully explored in requirements elicitation, analysis and specification for CAS.

3. A common architecture for CAS

CAS are conceptually characterized by the fact that, apart from the *basic functionality – a.k.a. system core*, the *system environment* and the *user interface* for bridging the gap between the system and its environment, they also contain an *adaption subsystem*, responsible for the enrichment of the basic functionality with further situational functionality [32]. Sometimes they may contain a *calibration subsystem* responsible for the adjustment of the adaptation logic to fit to new environment conditions [11]. These conceptual parts of CAS are semantically correlated as illustrated in figure 1.

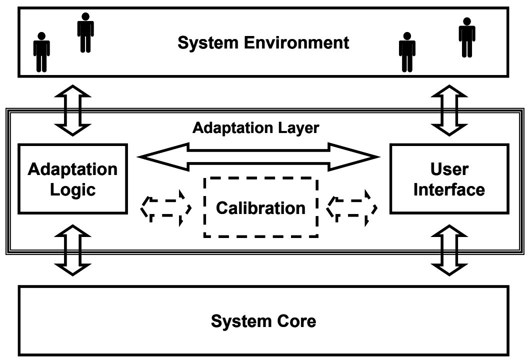


Figure 1. A multi-layered architecture

System Core: It represents the minimal necessary system functionality. Such functionalities are situation independent and their availability is always guaranteed. The system will therefore be able to provide these functionalities in any possible usage situation.

Adaptation Logic: It aims at bridging the gap between the system and its environment including the user. Thus, it is responsible for the communication between both worlds. The adaptation logic can be regarded as a universal filter mediating any kind of communication observable at the interfaces between the system and its environment. It is composed of:

a) User Interface: The UI is an essential element in each interactive systems. It represents all aspects of the system which can be seen (heard or otherwise perceived) by the human user, and the commands and mechanisms the user may utilize to control its operation and input data. In CAS, the UI is responsible for any direct (sometimes also conscious) communication between the user and the system.

b) Adaptation: It realizes the adaptation logic needed for the enrichment of the basic functionality. The decision, whether and how an adaptation of the system is to be accomplished and under which conditions it should take place, is specified in the adaptation logic.

c) Calibration: This optional part of the system is responsible for the retrospectively adjustment of the adaptation logic. It is often defined as a manual adaptation of the adaptation logic. Manual because it requires additional user expertise, and it is not stimulated by the system itself.

System Environment: It is source of important information needed to characterize the system's situation. Fluctuating components used to extend systems functionalities are contained in the environment.

4. The RE-CAWAR methodology

RE-CAWAR is a model-based approach to RE for CAS. It aims at eliciting, analyzing and specifying the different conceptual parts of CAS as presented in section 3, starting from the user and business needs. Thereby the proposed approach explicitly elaborates an integrated model of the usage context (operational context or context of use) especially needed for elicitation and system scoping (setting system's boundaries and limitations).

4.1. Integrated model of usage context

In ubiquitous computing, the notion of context is often equated with location. In fact there is more to context than location. Tarasewich argues that context is actually complex by analyzing the concept from different perspectives [30]. He considers almost all appreciable taxonomies including Schilit et al. [26], Schmidt et al. [28] and Dey [9]. He states that aspects such us participants, activities and environments should not be neglected in context modeling. Time may also play an important role in that taxonomy.

The integrated model of the usage context in RE-CAWAR is quite close to Tarasewichs taxonomy. We distinguish three dimensions of the usage context and model their interrelated change over time (fig. 2). The time aspect is important for prediction issues in context-awareness.

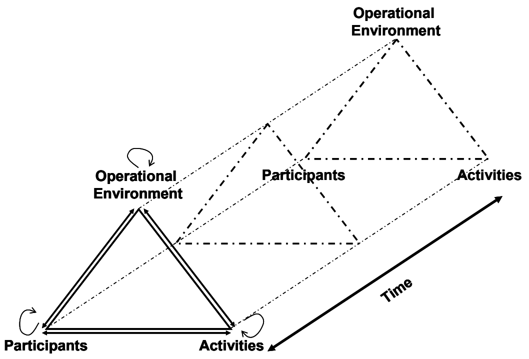


Figure 2. Context as dynamic construct

Changing Participants: This dimension encloses aspects regarding location and orientation of participants, their personal properties (e.g. age, education), their mental (e.g. mood, anger, stress) and physiological states (e.g. pulse, blood pressure), their personal expectations and their social dependencies.

Changing Activities: This dimension concerns tasks and goals of participants influenced by events in the environment (e.g. weather is fine, so go to work by bicycle).

Changing Operational Environment: This dimension includes aspects such as location of the application, network conditions, devices and communications quality and availability, orientation of entities, physical factors such as temperature, light, humidity and noise.

A thorough exploration of these different aspects of context resulted in the necessity to provide a systematic for context handling. We therefore provide an integrated model for the usage context. This integrated model as such consists of the user model, the task model, and the domain model representing the tree aspects of the usage context, and orthogonally also the platform model, the dialog model and the presentation model supporting the early three models.

- *The User Model* represents the participants aspect. It characterizes the users and the user groups. Stereotypes are often included in the user model.
- *The Task Model* represents the activities aspect. The task model is responsible for identifying which task and which interactions are needed to perform it.
- *The Domain Model* represents the operational environment aspect. It consists of any user visible, -accessible and -manipulable objects in the applications domain.
- *The Platform Model* represents the physical infrastructure and the relationship between the involved devices.
- *The Dialog Model* represents the interaction between user and system.
- *The Presentation Model* represents visual, haptic and audio elements needed for the interaction.

All these models serve as an integrated template for eliciting requirements not only for the system core and the UI, but also for the identification of contextual (situational) needs. These needs are to be automatically identified by the system at runtime [31, 4] leading to an adaption of the system. The models may also serve as checklists for preserving from omitting important aspects of the usage context.

4.2. The core of the methodology

RE-CAWAR iteratively specifies context adaptive systems starting from the user's and business needs and integrates them from the beginning on (fig.3). It is based on approved methods from traditional and model-based RE such as scenario- and goal-based approaches. These methods are enriched with further methods from Usability Engineering and User Modeling. Doing so as much as possible usage situations are made foreseeable for the system while keeping the user informed about the limitations of the system's use. Since the calibration of a system always requires additional interaction and to some extent also a technical expertise [13], a reduction of the calibration subsystem to the absolute minimum is aimed at.

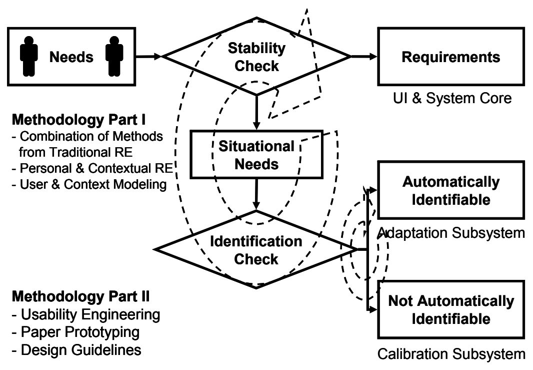


Figure 3. An overview of RE-CAWAR

Part I: Stability Check

The main objective of this part consists of eliciting, analyzing and specifying requirements for both the *system core* and the *user interface*. Scenario-based approaches are considered to be very promising methods in the development of systems with expected changing context at deployment time [2, 17]. They are expected to bridge the gap between users and requirements engineering. We absolutely confirm this expectation based on own experiments in the CAWAR research group during the last seven years, where we developed several prototypes for context aware and ubiquitous applications. Based on the developed templates for the usage context, we recommend to formulate scenarios to determine the basic functionality of the system, and its interfaces. RE done at this stage correspond to the level 1 RE defined in [4], where the basic functionality of the system is determined. During the development of scenarios the gained insight information is used to enrich the integrated model of the usage context. Goal-based approaches should help to develop the tasks that are to be performed using the system. We recommend the GRAM approach [3]. To model tasks concur task tree (CTT) and derivate are common [23]. CTTs are in fact very close to goal graphs, so their use in conjunctions with goal-based approaches should be straight forward. User modeling techniques [16] may help to develop solid user models needed to represent the participant aspect of the usage context. For the elaboration and development of the other models integrated into the model of usage context, we actually recommend the use of scenario-based approaches as described above. The PC-RE approach [29] for instance should be helpful to capture elements belonging to the domain model. After developing all these models, one should integrate them by means of requirements chunks, thus characterizing which requirements are obviously common to all or most instances of the usage context. These are extracted from the stable needs (needs that remain valuable in several usage context). After this stability check, needs that are classified as strongly depending on the usage context (situational needs) are gathered. They serve as input for the second part.

Part II: Identification Check

This part aims at identifying needs that could be automatically recognized by the system at runtime and at converting them into adaptation requirements. RE done at this stage is to prepare the level 2 RE defined in [4], where the system should determine which adaptation is appropriate for a given situation. Also adaptation elements, that will allow the system to adapt to changing conditions, are to be defined in this part of RE-CAWAR. It corresponds to level 3 RE in [4]. Making things automatic (as often done by context adaptation), it is very likely that usability problems occurs. Therefore we recommend to highly consider usability aspects in this part of the methodology. Usability methods such as contextual inquiry, focus groups, use case definition and card sorting may be helpful. Principles such as task analysis and user analysis, known from usability engineering, should be integrated into the analysis activities, specially into activities related to the problem understanding, modeling of the context of use and also the elicitation. This is also valuable for the first part of RE-CAWAR. Furthermore, synchronization approaches based on the principle of learning are pursued. On the one hand the system will explain the adaptation behavior to the user, e.g. by providing appropriate feedback to the user. On the other hand certain adaptations might not occur, since the differentiation of the situation is not obvious, otherwise the user will only be confused. Last but not least, guidelines based on the principle of understanding and the execution should be considered. Carefully designed feedback over possible and actual information flow may help the user to better understand the behavior of the system after an adaptation. Early prototypes may also help.

5. Proof-of-concept

As mentioned in the introduction of this paper, a prototype system was developed for evaluating the introduced concepts. It is about an autonomous task scheduler (the **C**ontext **A**ware **T**ask **S**cheduler). Appointments, events and other personal tasks of a user can thereby be imported into the application context from different sources such as email, online calendar or even public event repositories.

The CATS furthermore provides some basic functionality for managing this data. Several forms of adaptive notification functionalities were implemented, which for instance inform about rearranged or conflicting schedules and remind users of upcoming appointments. The type of a notification is context aware: the application for instance realizes if the user is currently in a meeting and in consequence informs him unobtrusively or not at all, if the notification is rated as less important. Besides the functionality of manually rearranging schedules or setting filter rules concerning unwanted events gathered from public repositories, a mechanism for the automatic rearrangement of timely overlapping events and appointments is provided. This example is not new (e.g. [24] or [35]). We chose this case study because it has been considered many times and therefore results can easily be compared.

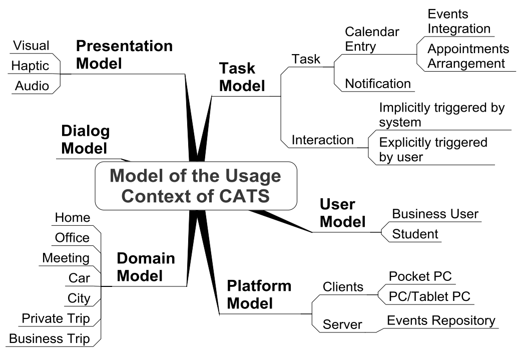
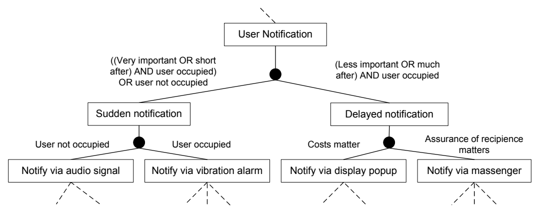


Figure 4. Initial model of the usage context

According to the RE-CAWAR-methodology the development of an integrated usage context is crucial for the development of context aware applications. This model is used as a checklist during the elicitation and analysis of requirements. Figure 4 illustrates an initial model of the usage context. During the elicitation and analysis of user needs the model may be enriched by further details. The integrated method may be useful, not only for the first part of RE-CAWAR where the systems basic functionality is determined, but also for the second part where the adaptation behavior of CATS is specified.

To satisfy the goal that the user may prefer to be notified silently anytime he is in a meeting, the system should not only be able to notify the user in a silent mode, but also it may be aware of the current usage situation (see figure 5). Table 1 summarizes corresponding requirements chunk.

The adaptive behavior of the system is modeled by meand of the K-Model[11]. It is used for a first tentative description of the functionalities of CATS. This model is capable of being iteratively enriched and refined by information gained from scenario analysis and prototyping.



User Notification
((Very important OR short after) AND user occupied) OR user not occupied
(Less important OR much after) AND user occupied
Sudden notification
Delayed notification
User not occupied
User occupied
Costs matter
Assurance of recipience matters
Notify via audio signal
Notify via vibration alarm
Notify via display popup
Notify via massenger

Figure 5. A cut-out of the goal graph of CATS

Requirements R12
use vibration alarm for user notification
Context C12
1. User is occupied. 2. Notification is due. 3. User can interact with the system at anytime.
Scenario Sc28
Mr. William is currently in a meeting as the cancellation of the planned press conference is published. According to the specified user profile, the CATS client classifies this event as interesting for Mr. William and decides to notify him. In order to not disturb other meetings participants, the system notifies him by means of silent mode.

Table 1. A cut-out of requirements chunk

6. Conclusion and further works

In this paper we presented the RE-CAWAR approach. RE-CAWAR is a methodology that aims at augmenting the RE process for context aware and adaptive systems. The core of the methodology is an integrated model of the usage context. Since context is more than location, the integrated model enriches the notion of context with the aspects of participants, activities and operational environment including their changes over time. An iterative process enables the identification of stable needs and such that may change according to the context. Stable needs result in the basic functionality of the system and are implemented in the system core and the user interface. The situational needs are further analyzed and used to specify the adaptation logic.

As next steps we will refine the methodology part one and part two. The integrated model was made up ad-hoc in past projects and will be systematized. A common interface will be defined to allow for seamless integration of new methods into the methodology.

References

[1] G. Abowd and E. Mynatt. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, 2000.

[2] C. B. Achour, C. Souveyet, and M. Tawbi. Bridging the gap between users and requirements engineering: the scenario-based approach. *Intl. Journal of Computer Systems Science and Engineering*, 14(6):379–406, 1999.

[3] A. Antón. *Goal Identification and Refinement in the Specification of Software-Based Information Systems*. PhD thesis, Georgia Institute of Technology, 1997.

[4] D. M. Berry, B. H. Cheng, and J. Zhang. The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems. In *11th Intl. Workshop on Requirements Engineering: Foundation for Software Quality*, 2005.

[5] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-Aware Electronic Tourist Guide: some Issues and Experiences. In *SIGCHI Conf. on Human Factors in Computing Systems*, pages 17–24. ACM Press, 2000.

[6] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In *19th Intl. Conf. on Software Engineering*, 2001.

[7] N. Davies, J. Landay, S. Hudson, and A. Schmidt. Rapid Prototyping for Ubiquitous Computing. *Pervasive Computing*, 4(4), 2005.

[8] A. De Angeli, U. Athavankar, A. Joshi, L. Coventry, and G. Johnson. Introducing ATMs in India: a Contextual Inquiry. *Interacting with Computers*, 16(1):29–44, 2004.

[9] A. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[10] A. K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[11] M. Fahrmair, W. Sitou, and B. Spanfelner. An Engineering Approach to Adaptation and Calibration. In T. R. Roth-Berghofer, S. Schulz, and D. B. Leake, editors, *Modeling and Retrieval of Context: MRC 2005*, volume 3946, pages 134 – 147. Springer, 2006.

[12] M. Fahrmair, W. Sitou, and B. Spanfelner. Unwanted behavior and its impact on adaptive systems in ubiquitous computing. In *ABIS 2006: 14th Workshop on Adaptivity and User Modeling in Interactive Systems*, Hildesheim, Germany, October 2006.

[13] M. R. Fahrmair. *Kalibrierbare Kontextadaption für Ubiquitous Computing*. PhD thesis, Department of Informatics, Technische Universität München, Germany, 2 2005.

[14] S. Fickas. Clinical Requirements Engineering. In *27th Intl. Conf. on Software Engineering*, St. Louis, May 2005.

[15] S. Fickas, W. Robinson, and M. Sohlberg. The Role of Deferred Requirements in a Longitudinal Study of Emailingl. In *13th IEEE Intl. Conf. on Requirements Engineering*, 2005.

[16] G. Fischer. User Modeling in HumanComputer Interaction. *User Modeling and User-Adapted Interaction*, 11(1-2):65–86, 2001.

[17] L. Kolos-Mazuryk, G.-J. Poulisse, and P. van Eck. Requirements Engineering for Pervasive Services. In *OOPSLA'05 Workshop on Creating Software for Pervasive Services*, 2005.

[18] L. Kolos-Mazuryk, P. A. T. van Eck, and R. J. Wieringa. A survey of requirements engineering methods for pervasive services. Deliverable TI/RS/2006/018, Freeband A-MUSE, Enschede, 2006.

[19] G. Kotonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley, John & Sons, 1998.

[20] D. A. Norman. *Emotional Design: Why We Love (or Hate) Everyday Things*. Basic Books, 2004.

[21] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. In *ICSE '00: Conf. on The Future of Software Engineering*, pages 35–46. ACM Press, 2000.

[22] C. Potts, K. Takahashi, and A. Anton. Inquiry-based Requirements Analysis. *Software, IEEE*, 11(2):21–32, 1994.

[23] C. Pribeanu, Q. Limbourg, and J. Vanderdonckt. Task Modelling for Context-Sensitive User Interfaces. In *8th Workshop of Design, Specification and Verification of Interactive Systems*, 2001.

[24] B. Rhodes. The wearable remembrance agent: A system for augmented memory. *Personal Technologies Journal Special Issue on Wearable Computing*, 1:218–224, 1997.

[25] S. Robertson and J. Robertson. *Mastering the Requirements Process*. ACM Press/Addison-Wesley, NY, USA, 1999.

[26] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.

[27] A. Schmidt. *Ubiquitous Computing - Computing in Context*. PhD thesis, Computing Department, Lancaster University, U.K., 2002.

[28] A. Schmidt, M. Beigl, and H. Gellersen. There is more to context than Location. *Computers & Graphics*, 23(6):893–901, 1999.

[29] A. Sutcliffe, S. Fickas, and M. Sohlberg. PC-RE: a Method for Personal and Contextual Requirements Engineering with some Experience. *Requirements Engineering*, Vol. 11(No. 4):157–173, 2006.

[30] P. Tarasewich. Towards a Comprehensive Model of Context for Mobile and Wireless Computing. In *Americas Conference on Information Systems - AMCIS*, 2003.

[31] M. Trapp. *Modeling the Adaptation Behavior of Adaptive Embedded Systems*. PhD thesis, University of Kaiserslautern, 2005.

[32] M. Trapp and B. Schürmann. On the Modeling of Adaptive Systems. In *Intl. Workshop on Dependable Embedded Systems, Italy*, 2003.

[33] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *5th IEEE Intl. Symposium on Requirements Engineering*, page 249. IEEE Computer Society, 2001.

[34] M. Weiser. The Computer for the 21st Century. *Scientific American*, 3(265):94–104, September 1991.

[35] J. Wohltorf, R. Cisse, A. Rieger, and H. Scheunemann. BerlinTainment: An Agent-Based Serviceware Framework for Context-Aware Services. In *MobiSys Workshop on Context Awareness*, 2004.

[36] E. S. K. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *3rd IEEE Intl. Symposium on in Requirements Engineering*, pages 226–235, 1997.