An Engineering Toolbox to Build Situation Aware Ambient Assisted Living Systems

Werner Kurschl, Stefan Mitsch and Johannes Schoenboeck
Upper Austria University of Applied Sciences
Research Center Hagenberg
Department of Software Engineering
Softwarepark 11, A-4232 Hagenberg, Austria
{kurschl, smitsch, jschoenb}@fh-hagenberg.at

Abstract

Due to increasing anticipated average life and health expenditure ambient assisted living (AAL) systems attract the attention of researchers. To successfully build and deploy AAL systems knowledge from different fields of computer science is needed: pervasive computing to gain the raw data, machine learning and pattern recognition to interpret these data and HCI knowledge to allow implicit interaction with the system.

In this paper we propose a reference architecture for building AAL systems. Based on this reference architecture we introduce a toolbox that simplifies the development of AAL systems. The toolbox consists of a meta-model for pipeline systems, a low-level context model, high-level context ontologies, customizable components and tool support.

1. Introduction

Routine medical check-ups often needed by elderly, handicapped, or otherwise needy people in hospitals or at the local doctor pose significant effort on both, the patient and the medical staff. Moreover, these check-ups are just a snapshot of the patient's physical constitution, because collecting continuous data would be too costly in terms of time and money. Additionally, patients often have to stay at hospitals just for the purpose of monitoring their well-being and to do some routine check-ups (e.g., after a medical treatment or a surgery). If such patients could leave hospitals earlier and do these routine check-ups themselves at home, the medical staff would be effectively unburdened from routine work, while at the same time the medical care quality could be enhanced by continuous monitoring ([22] also emphasizes the importance of continuous monitoring).

To reach these goals we envision ambient assisted living (AAL) systems that consist of various hardware and software components integrated into everyday items or worn/used by patients. The collected information could be evaluated—either by medical staff or by an AAL system—for irregularities, alarming changes, or symptoms. The system could even automatically alert relatives, care givers, or medical staff (taking into account a regulatory framework for processing medical data as well as privacy protection).

Another aspect of an AAL system is to improve elderly and needy persons' habitability: it should assist them in living autonomously, and let them participate in social communities and family life. Envisioned living assistants include item tracking and searching, warning of household dangers (e.g., slippery floor, unattended stove, or running water taps), and recognition of alarming situations (e.g., collapse, sleep disorders). Social applications include communication, entertainment, and awareness displays (e.g., neighbor at home, or family emotions).

To recognize these situations sensors integrated within the living environment are needed. To date hardware sensors and software components that can provide or interpret some of the data needed already exist. But a common infrastructure to integrate these components has yet to be developed. AAL systems require a stable and profound architecture, which enables the seamless integration of different sensor types to acquire quantitative raw data. A situation inference engine transforms quantitative low-level context into qualitative high-level context data (i.e., situations). These situations are the basis to reason about possible actions (e.g., update a display, switch an actuator, or call the ambulance).

In this paper we present a reference architecture for AAL systems and propose a development toolbox to simplify the implementation of such systems using a Model Driven Software Development (MDSD) approach.

2. AAL System Reference Architecture

Baldauf et al. compare in [4] various architectures in their survey on context-aware systems: (i) direct sensor access applications tightly integrate sensor access and application, which is useful for small, stand-alone applications, (ii) middleware-based applications use layers to separate low-level sensor details from application details, and (iii) context-server applications additionally permit multiple clients to access shared context data on a server.

Context-server applications, mainly used for distributed systems, are most flexible because heterogeneous clients and sensors can be integrated. Additionally, resource-intensive operations can be transferred from power-restricted sensors to a server. SOCAM (see [14]) and JCAF (see [6]) are examples of context-server applications. [2] and [31] describe how a layered architecture can separate detecting and using context to improve extensibility and reusability.

Figure 1 shows the reference architecture we use throughout this paper (based on the described work). We structure the architecture into three main tiers according to a typical deployment's hardware boundaries. Within each tier an n-layer (e.g., data layer, business layer, and presentation layer) design might prove useful.

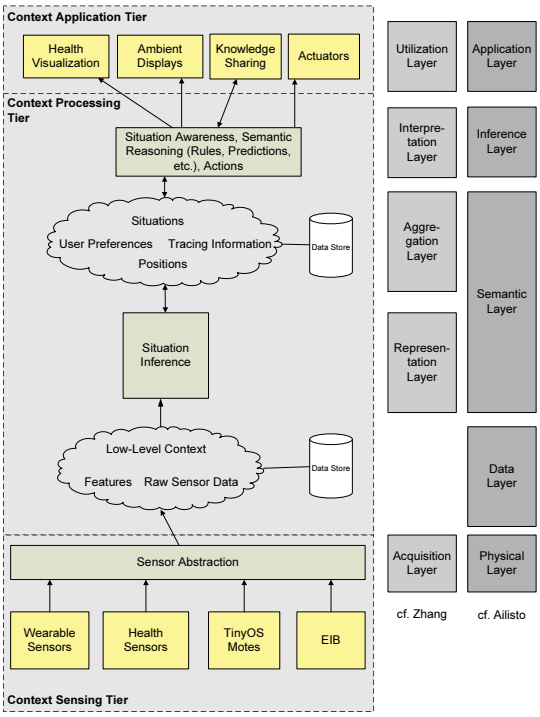

Figure 1. AAL System Reference Architecture with Layers according to [31] and [2]

2.1. The Context Sensing Tier

Ambient assisted living applications often need different types of measurements or sensor values (e.g., sound, acceleration, water flow, light) from the surrounding environment, which are usually provided by sensors and processing boards from different manufacturers. To ease the integration of these sensors a sensor abstraction layer is necessary, which uses common plug-in software patterns for sensor-specific implementation parts. The data provided by the sensor abstraction layer is furthermore described in terms of a common data model, thereby hiding sensor-specific issues within the sensor abstraction layer.

2.2. The Context Processing Tier

The context processing tier provides persistent storage for context data. Moreover it derives high-level context (situations) from low-level context (raw context data from the sensors) using feature extraction, machine learning, and pattern recognition algorithms. These high-level context data, if described in terms of an ontology, are the foundation for situation awareness as defined by [12]. Reasoning on situations permits to assess situations and predict future developments (situation evolution).

2.3. The Context Application Tier

The context application tier contains applications that utilize context information to adjust their behavior. Different applications might listen to events of the reasoning engine including explicit and implicit systems as mentioned in [21]. Explicit interaction systems always require a kind of dialog between the user and a particular system or computer. Especially for AAL systems implicit human interaction should be taken into consideration. Schmidt defines implicit human-computer interactions as interaction of a human with the environment and with artifacts aiming to accomplish a goal. The output of implicit systems should be seamlessly integrated with the environment of the user. [13] proposes to use ambient displays that operate at the periphery of a user's attention (e.g., some artwork).

Furthermore different types of actuators could be consumers of events of the reasoning engine. Knowledge sharing with other systems (e.g., medical information systems) may also be implemented here.

3. Development Toolbox Architecture

To support the development of AAL applications that adhere to the reference architecture described above, we propose a development toolbox, which consists of

- a meta-model for pipeline systems,
- a low-level context model,
- high-level context ontologies,
- customizable components (sensor abstraction, feature extraction, classifier learning algorithms, etc.)
- and modeling and development tools

for each of the three application tiers. The toolbox should assist developers in the following activities: software development for (i) sensing devices, (ii) the data processing parts, and (iii) data analysis and classification to identify situations.

3.1. Meta-Model for Pipeline Systems

For implementing the context processing tier the toolbox provides components for feature extraction, supervised classifier learning, and unsupervised clustering (all from the field of machine learning and pattern classification). The processing of raw data, which is typically provided by sensors as data stream, is done in a processing chain (see design pattern Tee-and-Join-Pipeline-System in [11]).

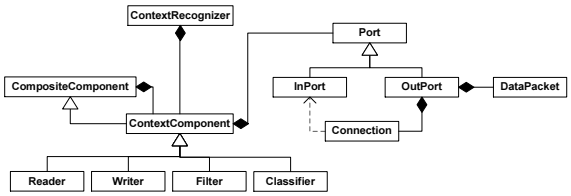

Figure 2. Meta-Model for Tee-and-Join-Pipeline-System Context Processing

The meta-model shown in Figure 2 defines the basic elements for modeling situation inference components. The class *ContextComponent* acts as an abstract base class for all components; concrete classes (not shown within the figure) provide specific behavior, e.g., a *MinFilter* component calculates the minimum value of incoming *DataPackets*. DataPackets cover the actual data delivered by a physical sensor. ContextComponents can be connected via *InPorts* and *OutPorts* using *Connections* that transfer data from one component to another. If components run on different hardware devices or processes specialized connections hide the specifics of remote calls. The *ContextRecognizer* class aggregates all ContextComponents and Connections within a certain model and thus forms the whole pipeline system.

The meta-model provides four different abstract base types of components, namely *Reader*, *Writer*, *Filter* and *Classifier*. A reader provides data from a specific source (e.g., a file or a sensor), whereas a writer forwards data to a specific sink (e.g., console, file, database or actuator). Filters extract features from raw sensor data. These features can be used in classifiers to recognize situations. *CompositeComponents* can be used to hierarchically group components. Thereby a Pipes-and-Filters chain modeled as CompositeComponent that recognizes a certain situation (i.e., a ContextRecognizer) could be the basis for more complex applications.

3.2. Low-Level Context Model

Current context-aware applications strive to interpret or aggregate data and thereby minimize raw data storage demands. But in medical applications detailed raw data information needs to be preserved for later inspection by medical staff. We therefore define a model for representing low-level context data.

[4] and [24] present surveys on context models reaching from simple key-value models to ontologies. We find the Aspect-Scale-Context Information (ASC) model described in [25] as part of the CoOL ontology and the context atom attributes listed in [4] to be a flexible and general basis for building a context model describing low-level context (i.e., raw sensor data). To enable supervised classification learning algorithms to work on raw data we extend the models to contain class labels and classification domains. Figure 3 shows the extended model including an instantiation example.

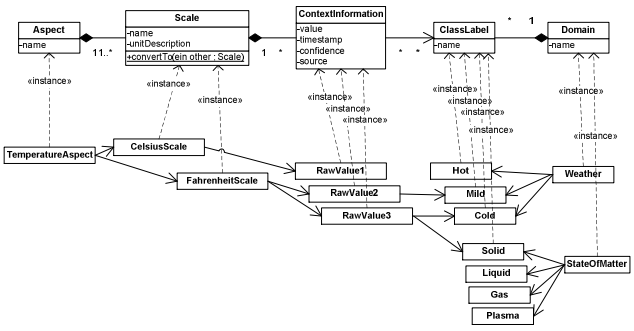

Figure 3. Low-Level Context Model

Each sample of Strang's (see [24]) context information (or context atom according to [4]) belongs to one aspect of the modeled knowledge in the system and can be interpreted according to a particular scale. Additionally, class labels may describe a context information sample in terms of semantics from a particular domain (thereby enabling supervised learning algorithms).

3.3. Situation Awareness and Ontologies

The envisioned AAL system produces a huge amount of raw sensor data and also interpreted high-level context. As Endsley puts it in [12]: "The problem with today's systems is not a lack of information, but finding what is needed when it is needed". He describes situation awareness as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future", which we agree to be a possible solution to this problem. In the sections above we described how to perceive the environment by using sensors, and how to represent raw sensor data in a common model. Feature extraction and classification algorithms then transforms raw data into high-level context (situations), thereby deriving semantic meanings. To be beneficial and applicable to situation awareness the derived situations must adhere to yet another model in order to relate and assess current situations, but also to predict future developments. [7] proposes the use of upper ontologies as modeling constructs for situation awareness in road traffic management, as they could permit the integration of information about perceived objects, identify situations, and share knowledge. [27] additionally names the possibility of logic inference and knowledge reuse as major reasons for using ontologies.

Numerous ontologies already exist in the pervasive computing domain: [7], [18], and [24] present surveys on ontologies for situation awareness and context modeling. [8], [19], and [9] elaborate the basic ideas (deriving additional meaning by reasoning about situations) towards prediction of future developments in the road traffic management domain. We consider these concepts also to be relevant and beneficial in AAL systems and other pervasive environments and will therefore include an appropriate ontology and tools into our toolbox. A common shortcoming we found among all described models is missing tracing information. As already stated above medical data typically needs to be available in raw format for detailed inspection by medical staff; thus, trace links from derived high-level context to raw data must be preserved to allow medical staff to quickly navigate to relevant data. We will therefore extend the existing work to also contain such tracing information.

3.4. Customizable Components

Many implementation tasks during development of the sensing or processing tier are recurring in different AAL systems. One approach for a reduced development effort is the use of a generic software framework or platform with customizable components. A customizable component is one that can be tailored for reuse in a new context prior to its installation or use (see [28]).

The meta-model described in section 3.1 is based on the Eclipse Modeling Framework (EMF) which represents the model in Ecore. Programmers are able to extend the provided model by creating a new ecore model that references the base model. Within the new model the user can easily integrate new ContextComponents or customized Connections.

The abstraction of different sensors and hardware platforms (e.g., Crossbow MICAz with TinyOS and ZigBee) is a critical issue for AAL applications. Various different manufacturers provide specialized sensors that allow programmers to gain specific context data. Thus the toolbox must ensure the seamless integration of different sensors and processing boards. TinyDB (see [17]) provides a query language for sensor networks similar to SQL with additional attributes like sample period and duration. On the one hand it simplifies querying sensor values and on the other hand it abstracts from different hardware platforms. That is why we integrated a *TinyDBReader* into our meta-model providing sensor data for further processsing in filters or classifiers.

A simple filter might be a minimum filter or a mean filter, a more complex one could be for example a Fourier-Transformation filter. As stated above filters provide features for classifiers. Currently we integrated machine learning and pattern recognition algorithms from WEKA (see [30]) into our meta-model. In the future we plan to integrate heuristic and genetic algorithms (see [29]).

3.5. Modeling and Development Tools

To further increase productivity appropriate tool support is needed. Based on the meta-model we provide a graphical editor for defining model instances. The graphical editor uses the Eclipse Foundations's Graphical Model Framework (GMF). The context components are presented within a palette of the editor. By using drag and drop instances of components can be included into the model. *CompositeComponents* can be used to split the model into several sub models. Attributes of context components can be edited within the properties view. For complex attributes custom editors are provided. Figure 4 shows the current prototypcal implementation of the graphical editor. The figure shows a simple model that identifies if a person is standing or walking. The model makes use of a reader for fetching the values from the orthogonal accelerometers of an appropriate sensor board (we call them x- and y-acceleration). As the sensor delivers constant values when the person is standing and unsteady values when the person is walking, a variance filter could be used to calculate a threshold which can be used in a simple classifier to predict the actual outcoming (standing or walking).

The model shown in figure 4 produces code for the

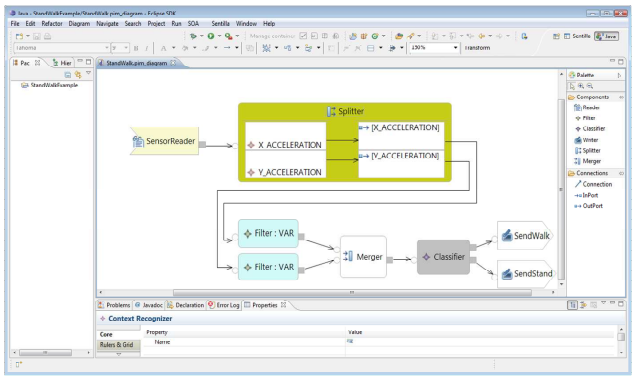

Figure 4. Graphical Editor Prototype

Eclipse platform. Currently we provide models for the Eclipse/Java and the TinyOS platform. To be conform to the MDA-approach we are going to split-up the model into a platform independent model (PIM) and a platform specific model (PSM). Thereby it should be possible to transform a PIM into one or more PSMs. After modeling an AAL system source code for different platforms can be generated. The EMF Validation framework is used to check additional constraints (e.g., all components must have a unique name). Figure 5 shows the template-based code generation process using the Xpand[1] template language. The code generator uses an instance of the meta-model and the template to generate source code for a specific platform. The components itself are provided as archive file implemented in a platform-specific language and are referenced by the generated project.

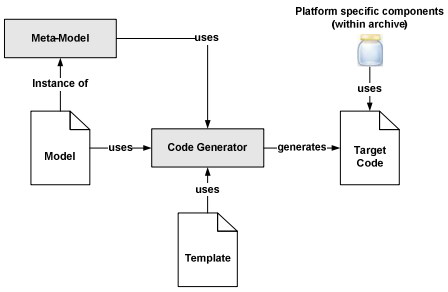

Figure 5. Code Generation

This approach of code generation supports also in-network processing and therefore reduces overhead, because processing and communication is shifted from the server-side to the wireless sensor network. In-network data processing can improve the scalability and can reduce the energy/resource consumption, since it may significantly reduce the data volume that has to be routed through the network (see [20]).

[1] www.eclipse.org/gmt/oaw/doc/4.1/r20_xPandReference.pdf

As stated above it should be possible to integrate various platforms into the toolbox. Thus it must be possible to easily extend the code generation process as well. As we use a template based approach a new template has to be created if source code for a new platform should be generated. We provide an extension point where a Java class has to be provided that implements the generation process and templates for a particular platform.

4. Related Work

SOPRANO is an EU-funded project aiming to assist older people resulting in a more independent life in their familiar environment. The main part is the SOPRANO Ambient Middleware (see [15]) which receives sensor inputs and user commands, enriches them semantically using ontologies and triggers appropriate reactions using different types of actuators.

BelAmi[2] is a project at the Fraunhofer IESE in the field of ambient intelligence systems. They argue that AAL systems raise a series of new challenges to software development due to the combination of mobility, adaptivity and resource scarceness, which could be solved by software product lines as proposed in [3].

Lymberopoulos demonstrates in [16] the application of a probabilistic grammar-based formulation to detect complex activities from simple sensor measurements. In particular, the authors present a grammar hierarchy for identifying cooking activity from low-level location measurements in an assisted living application.

The Context Recognition Network (CRN) Toolbox (see [5]) describes a C++ framework integrating hardware abstraction, filter algorithms, feature extraction components and classifiers in a configurable runtime to support rapid development of context recognition applications. It is designed for deployment to embedded devices that support the POSIX runtime environment. Although a graphical editor is provided it is not based on a formal meta-model and thus the CRN Toolbox cannot fully benefit from a MDSD approach. As we base our toolbox on a formal meta-model we are able to generate code for various platforms, including the CRN Toolbox, by exchanging the generator templates.

Model Driven Software Development (MDSD) is a software paradigm where software is (partly) generated from models (see [23]). The Eclipse Foundation, in particular the Eclipse Modeling Framework (EMF) (see [10]) and the Graphical Modeling Framework (GMF) (see [26]), provides rich support to easily implement models and (graphical) editors needed for an MDSD approach.

WEKA (see [30]) provides a collection of machine learning algorithms for data mining tasks fully implemented in

[2] www.belami-project.org/index.html

Java. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

MATLAB [1] is a numerical computing environment and programming language. MATLAB provides among other things a so called Neural Network toolbox (accumulation of several small scripts implemented in a proprietary programming language) for designing and simulating neural networks used for pattern recognition. Although MATLAB would provide powerful mathematical operations for data analysis it can hardly be integrated into a complete software system. Nevertheless MATLAB could serve as a possible platform for code generation (e.g., for simulation using SIMULINK).

5. Conclusion and Further Work

In this paper a reference architecture for AAL systems was introduced. We described the major parts of the architecture and related our reference architecture to existing proposals. Based on the reference architecture we introduced a toolbox that simplifies building AAL systems. The toolbox provides basic components that read data from different sources, extract features, enable pattern recognition and write results to different sinks. By allowing programmers to extend our meta-model custom components can be integrated. To further increase productivity appropriate tool support is needed, which has been shown in a first prototype of the toolbox.

The toolbox must now be further evaluated in different AAL applications and different use cases to measure its advantages and shortcomings. Currently we focus on integrating different classifiers and define a suitable ontology for semantic reasoning. To be conform to the MDA specification we are going to split-up our model into a platform independent model (PIM) and a paltform specific model (PSM) and introduce model transformations.

Acknowledgment: This research was supported by the Upper Austrian state government. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily represent the views of the research sponsors.

References

[1] MATLAB. http://www.mathworks.de/. last accessed 05.2008.

[2] H. Ailisto, P. Alahuhta, V. Haataja, V. Kyllönen, and M. Lindholm. Structuring Context Aware Applications: Five-Layer Model and Example Case. In *Proceedings of Ubicomp 2002, Concepts and Models for Ubiqutious Computing Workshop*, 2002.

[3] M. Anastasopoulos, T. Patzke, and M. Becker. Software product line technology for ambient intelligence applications. Research Report, 2005.

[4] M. Baldauf, S. Dustdar, and F. Rosenberg. A Survey on Context-Aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2004.

[5] D. Bannach, O. Amft, and P. Lukowicz. Rapid Prototyping of Activity Recognition Applications. *IEEE Pervasive Computing*, 7(2):22–31, 2008.

[6] J. E. Bardram. Design, Implementation, and Evaluation of the Java Context Awareness Framework (JCAF). April 2005.

[7] N. Baumgartner and W. Retschitzegger. A Survey of Upper Ontologies for Situation Awareness. In *Proceedings of the International Conference on Knowledge Sharing and Collaborative Engineering (KSCE '06)*, 2006.

[8] N. Baumgartner, W. Retschitzegger, and W. Schwinger. Lost in Space and Time and Meaning – An Ontology-Based Approach to Road Traffic Situation Awareness. In *Proceedings of the 3rd Workshop on Context Awareness for Proactive Systems (CAPS 2007), Guildford, United Kingdom*, 2007.

[9] N. Baumgartner, W. Retschitzegger, W. Schwinger, G. Kotsis, and C. Schwietering. Of Situations and Their Neighbors: Evolution and Similarity in Ontology-Based Approaches to Situation Awareness. In *Proceedings of the 6th International and Interdisciplinary Confernce on Modeling and Using Context (CONTEXT 2007), Roskilde, Denmark*, 2007.

[10] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose. *Eclipse Modeling Framework*. Addison Wesley, 2004.

[11] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture. A System of Patterns*. John Wiley & Sons, 1996.

[12] M. R. Endsley. *Situation Awareness Analysis and Measurement*, chapter Theoretical Underpinnings of Situation Awareness: A Critical Review, pages 1 –24. Lawrance Erlbaum Associates, 2000.

[13] A. Ferscha. Informative Art Display Metaphors. In *Proceedings of the 4th International Conference on Universal Access in Human-Computer Interaction (UAHCI 2007)*, volume 4555, pages 82–92, Beijing, China, July 2007. Springer LNCS.

[14] T. Gu, H. K. Pung, and D. Q. Zhang. A service-oriented middleware for building context-aware services. In *Journal of Network and Computer Applications*, volume 28, pages 1–18, January 2005.

[15] M. Klein, A. Schmidt, and R. Lauer. Ontology-Centred Design of an Ambient Middleware for Assisted Living: The Case of SOPRANO. In T. Kirste, B. König-Ries, and R. Salomon, editors, *Towards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments (AIMCU), 30th Annual German Conference on Artificial Intelligence (KI 2007), Osnabrück, September 10, 2007*, 2007.

[16] D. Lymberopoulos, T. Teixeira, and A. Savvides. Detecting Patterns for Assisted Living Using Sensor Networks: A Case Study. In *Proceedings of 2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, 2007.

[17] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. volume 30, pages 122–173, New York, NY, USA, 2005. ACM.

[18] T. S. Reto Krummenacher. Ontology-Based Context Modeling. In *3rd Workshop on Context Awareness for Proactive Systems*, 06 2007.

[19] W. Retschitzegger and N. Baumgartner. Towards a Situation Awareness Framework Based on Primitive Relations. In *Proceedings of the Conference on Information, Decision, and Control (IDC'07), Adelaide, Australia*, 2007.

[20] K. Römer. *Time Synchronization and Localization in Sensor Networks*. PhD thesis, Swiss Federal Institute of Technology Zürich (ETH Zürich), 2005.

[21] A. Schmidt. *Ambient Intelligence*, chapter Interactive Context-Aware Systems - Interacting with Ambient Intelligence, pages 159 – 178. IOS Press, 2005.

[22] E. Shih, V. Bychkovsky, D. Curtis, and J. Guttag. Continuous medical monitoring using wireless microsensors. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 310–310, New York, NY, USA, 2004. ACM.

[23] T. Stahl and M. Völter. *Modellgetriebene Softwareentwicklung*. dpunkt.verlag, 2005.

[24] T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004, Nottingham, England, September 7, 2004*, 09 2004.

[25] T. Strang, C. Linnhoff-Popien, and K. Frank. CoOL: A Context Ontology Language to enable Contextual Interoperability. In J.-B. Stefani, I. Dameure, and D. Hagimont, editors, *LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, volume 2893 of *Lecture Notes in Computer Science (LNCS)*, pages 236 – 247, Paris/France, November 2003. Springer Verlag.

[26] M. Voelter, A. Haase, S. Efftinge, and B. Kolb. Graphical modeling framework. *iX*, 12:17, 2006.

[27] X. Wang. Ontology-based context modeling and reasoning using owl, 2004.

[28] R. Weinreich and J. Sametinger. Component models and component services: concepts and principles. pages 33–48, 2001.

[29] S. Winkler, M. Affenzeller, and S. Wagner. Advances in Applying Genetic Programming to Machine Learning, Focussing on Classification Problems. In *Proceedings of the 9th International Workshop on Nature Inspired Distributed Computing NIDISC '06, part of the Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium IPDPS 2006*, 2006.

[30] I. Witten and K. E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition edition, 2005.

[31] D. Zhang, T. Gu, and X. Wang. Enabling context-aware smart home with semantic web technologies. *International Journal of Human-friendly Welfare Robotic Systems*, 6:9, 2005.