2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems

Domain Independent Architecture and Behavior Modeling for Pervasive Computing Environments

Martin Peters, Christopher Brink, Sabine Sachweh
University of Applied Sciences and Arts Dortmund
Department of Computer Science
44227 Dortmund, Germany
{*martin.peters* || *christopher.brink* || *sabine.sachweh*}*@fh-dortmund.de*

***Abstract*—Although pervasive computing and ambient intelligence are emerging disciplines, domain independent middleware for those scenarios are rare. In this paper we propose act-mobile as a domain independent middleware for pervasive computing environments that consists of a central service platform and locally integrated gateways, building the bridge between sensors and actuators and the act-mobile server. Through the use of a centralized server we are able to connect to controlled environments over the internet and use different clients like mobile applications for monitoring and controlling. One key feature in achieving the domain independence is the way we define the behavior of the whole system and abstract from technical details to easily provide domain dependent functionality. The used statechart-based behavior modeling is also introduced in this paper. We show how we first build properties and composed properties to achieve a higher abstraction of sensor data and apply that abstraction to actuators, too. Finally, the rules are defined based on statecharts using composed properties and activities.**

***Keywords*-pervasive computing; middleware; behavior modeling; statecharts; domain independent;**

I. Introduction

Recent research activities in the field of ambient intelligence (AmI) and pervasive computing have a strong focus on the domain of home automation and ambient assisted living [1]–[4]. To date, many studies where performed in this field about different aspects like communications, context awareness or user interfaces and results were implemented in different scenarios like Health Care, Future Home or intelligent meeting rooms and offices [5]–[8]. Finally the objectives of an intelligent environment interacting with humans and considering context information are not limited to the aforementioned domains. Instead, AmI combines different areas of engineering and research such as artificial intelligence, robotics, wireless sensor networks and human-computer interaction which can also be applied to industrial installations to build an intelligent remote control and surveillance infrastructure.

The integration of an pervasive computing infrastructure into an industrial installation is challenging in different ways. Besides the circumstance that most industrial infrastructures do not originally have a centralized data store and fact basis to compute the current state of a facility to derive further activities, they first of all have to deal with a large number of heterogeneous sensors and actors. To be able to implement an intelligent environment, the protocols of such sensors and actors need to be mapped to a canonical form to establish an adequate way of communication between different system components. Because of the heterogeneity and the resulting complexity of such systems, especially solutions in the industry are domain specific and thereby expensive to implement. That is why complex solutions for remote surveillance and control combined with intelligent behavior typically are available for large-scale facilities only.

Another challenge in implementing such intelligent systems is the measurement of the current state of a facility including context information in an accurate way and to define the behavior of the whole system. While different approaches exists for context reasoning and decision making [9]–[12] they do not provide an abstract view, so that the behavior could be modeled by a technical layman. Even those approaches that allow a graphical modeling require a deeper understanding of the underlying facility because of the use of basic sensor and actuator values.

In this paper we introduce our pervasive computing platform called act-mobile, which focuses on providing a flexible and domain-independent infrastructure which is configurable and adaptable and can hence be used in different scenarios. The platform is based on a cloud application so that all information about a monitored facility are stored centralized and can be accessed from the internet. This allows us to provide different interfaces and use for example smartphones to monitor and control an installation. Further we introduce the configuration system to model the behavior of the act-mobile system, which is based on a multiple-step approach and therefore can not only be used by technical engineers, but also by people knowing the business processes which are performed by a system. The first step in behavior modeling is performed by a technical engineer who interprets data of single sensors and actuators and so provides a more abstract and natural view to the system while the next steps are based on that abstraction and allow a composition of multiple sensors and actuators. Finally we use statecharts [19] to define rules which can be simple and powerful at the

978-0-7695-4687-2/12 $26.00 © 2012 IEEE
DOI 10.1109/CISIS.2012.28

same time due to the aforementioned abstraction.

In the next section we start with an introduction of the act-mobile platform and describe the infrastructure design that allows us to use the platform domain-independent as well as the different aspects addressed by act-mobile. We also show how pervasive computing can be applied to industrial installations for intelligent surveillance and controlling. In section III we give a closer look into the process and implementation of the behavior modeling in act-mobile which will be the focus of this paper. Section IV will address related work and we are going to discuss the proposed work. The following sections will include an overview of further work and a conclusion.

II. The *act-mobile* platform

While the disciplines of ambient intelligence and pervasive computing overlap in a wide area, ambient intelligence mainly concerns the use of sensors and actuators integrated in a networked environment to provide a seamless and intuitive way of interaction with users, for example in a networked home or office. The term pervasive computing on the other hand is usually used in the context of industry and also describes the penetration of sensors and actuators in business processes. Both applications need to consider similar challenges and circumstances like heterogeneity, dynamically changing communication partners, processing of information, decision making, etc. Accordingly, different approaches and architectures have been proposed to build a middleware for ambient intelligence and pervasive computing [13], [14].

A drawback on most of the existing middleware infrastructures is that they are domain specific and do not allow the use in different scenarios. Especially in the industry this leads to expensive solutions that need to be developed and adapted to particular requirements. To address this issue we developed the act-mobile platform which is not only domain-independent, but also tries to address the availability of data, heterogeneity, extensibility and further concerns. In the following we are going to describe the architecture of act-mobile.

A. Architecture

The architecture of act-mobile is mainly based on a message oriented middleware which offers an easy way to extend the functionality of the system and to implement domain specific behavior. In contrast to other architectural paradigms like service oriented architectures, agent based architectures or aspect oriented implementations, the message orientated approach may be not as flexible as the aforementioned solutions but instead offers other advantages like loose coupling. To achieve loose coupling between the components and the different system parts, we use message queues which can be used by multiple components and communication partners simultaneously. Through

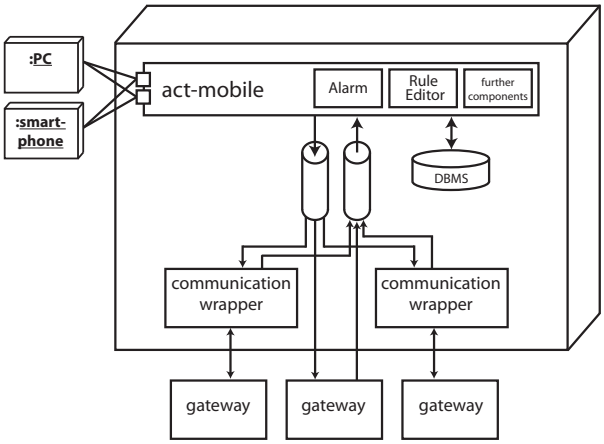

Figure 1. act-mobile architecture

this architecture style we can implement different message consuming components like alarm or logging components which will be used in most scenarios. Other components like e.g. for visualization, statistics or learning can simply be added depending on the individual needs and are able to read messages from the queue if necessary. Alternatively a statistics component may rely on the data store holding the complete history of states of a facility among other information. Another characteristic which distinguishes act-mobile from most other AmI architectures is the use of a centralized server which is able to support multiple clients and therefore multiple controlled environments, independent from the kind of environment which is connected to the system. This allows an efficient use of the hardware as well as the deployed software since multiple clients can share one architecture. Another advantage of the implemented client-server paradigm is that all data to a monitored facility as well as the connection to a facility to perform operations on it, can be accessed online. Through the use of different interfaces like RMI and HTTP offered by the system, different clients can be implemented like a website or smartphone apps. This allows platform-independent processing and analysis of the data as well as specific applications for individual requirements.

B. Gateway and connection management

The use of a cloud service as data store and access point for all kinds of operations allows us to implement small footprint gateways which are integrated in the monitored facility. Accordingly a gateway is used on the one hand to communicate with sensors and actuators while on the other hand it establishes the communication with the act-mobile cloud and synchronizes application data. Like depicted in figure 1 the communication can either be a direct communication through the message queues or be wrapped by a communication wrapper if the direct communication can

not be established due to the connection and implementation characteristics of a gateway. A communication wrapper may also be used to hide heterogeneity-specific communication protocols. The implementation of a gateway may differ depending on the scenario it is used in. For example a gateway may be implemented to be used for a small sewage treatment system which is typically not connected to the internet but requires regular monitoring and controlling. To connect such a system to the act-mobile platform and to establish the connection, a GSM module could be used which is limited regarding its connection bandwidth and is not capable to directly communicate through the message queues. Instead a communication wrapper can be used which builds a bridge between a specific gateway and the act-mobile system. It transforms messages from the gateway into the internally used format and vice versa. In other scenarios like the networked home where a permanent connection can be assumed, a direct communication through the message queues can be established. To achieve fault tolerance regarding the connection to the act-mobile cloud, the gateway has to provide an temporal data store, too, which fulfills different tasks. On the one side it is used to store data if no connection can be established so that a synchronization can be performed at a later date. On the other hand it can be used to filter and compress data, for example if a sensor provides continuous data. The gateway is also responsible for recognizing the current state of a facility and to provide the intelligence of an environment derived by the defined behavior that is described in section III.

III. Behavior modeling

To make an environment consisting of sensors and actuators responsive and intelligent and to provide a link between algorithms and the real world, different tasks need to be considered. These include the modeling of the environment, behavior modeling, activity recognition, decision making and reasoning to react on events. To fulfill this tasks, different approaches for behavior modeling and decision making exists which reach from the use of event-condition-action (ECA) rules [9], [15], [16] to the use of fuzzy rules [17] or neural networks [18]. While fuzzy rules are a good way to compensate impreciseness in the measurement of real world parameter and neuro-biological models are suitable for an iterative learning process, ECA rules are first of all easy to understand. An ECA rule basically exhibits the following format:

ON *event* IF *condition* THEN *action*

With this format it is straight forward to develop basic rules which trigger on an event (which may even be assembled by multiple events) and perform a defined action. However, in an ambient intelligence environment where a huge amount of sensors and actuators may exists which values may be not as easy to interpret the task of behavior modeling becomes a challenge. Considering a scenario where a small sewage treatment system is controlled by a rule-based system which ensures that a fuel level of a container does not exceed a specific level. A simple rule for this could be the following:

ON *s1.changed* IF *s1.value* $>$ *700* THEN *v1.set(1)*

The aforementioned rule triggers on the change of *s1* which is a fuel level sensor providing values between 0 (empty) and 800 (full) and opens a valve which accepts a *0* if the valve should be closed and a *1* if it should be opened. While the use case of this example is pretty easy (if fuel level to high, open valve), the according rule becomes complex because of the technical details like the interpretation of the appropriate values within this rule. Because of this circumstance, an easy rule which could be modeled by a domain expert without technical knowledge about the sewage treatment system must be developed by a technical engineer, which in turn may not have the necessary knowledge about the biological process. This problem might become even more sophisticated if whole business processes are monitored through a rule based pervasive computing architecture.

To address the aforementioned problems and to provide a bridge between the technical aspects and the business- and process-view of an intelligent system, we propose an approach based on three steps. We first design properties and composed properties of a system where composed properties consist of orchestrated properties and represent the state of an environment. In the second step activities (and also composed activities) are designed that can be executed in an environment. The third step connects the previous two steps and builds rules to model the behavior of a system based on statecharts.

A. Properties and abstract properties

Like depicted in figure 2 a), the modeling process of properties is tiled into three levels: the technical level (TL), ground level (GL) and business level (BL). The technical level is used to abstract from raw sensor data and to normalize the data into an appropriate format for further processing. This layer is also a key factor to address the heterogeneity of sensors and actuators in the act-mobile system. Each sensor is represented at this level independent from either being continuous or discrete while no interpretation of data is performed. The sensor data in turn is used to build properties which may be based on multiple sensor values which are connected through a diamond-symbol and evaluated through an expression, which must result in a boolean expression. The condition at the transition allows to compromise sensor values to define a properties state. On top of the technical layer the ground layer is used to define properties which are defined as boolean functions and accordingly can either be true or false. The state of a property results from the condition which is located at the transition between a sensor and the property so that the ground layer is used to completely abstract from technical details like values of specific sensors. Instead, properties like *high temperature* or

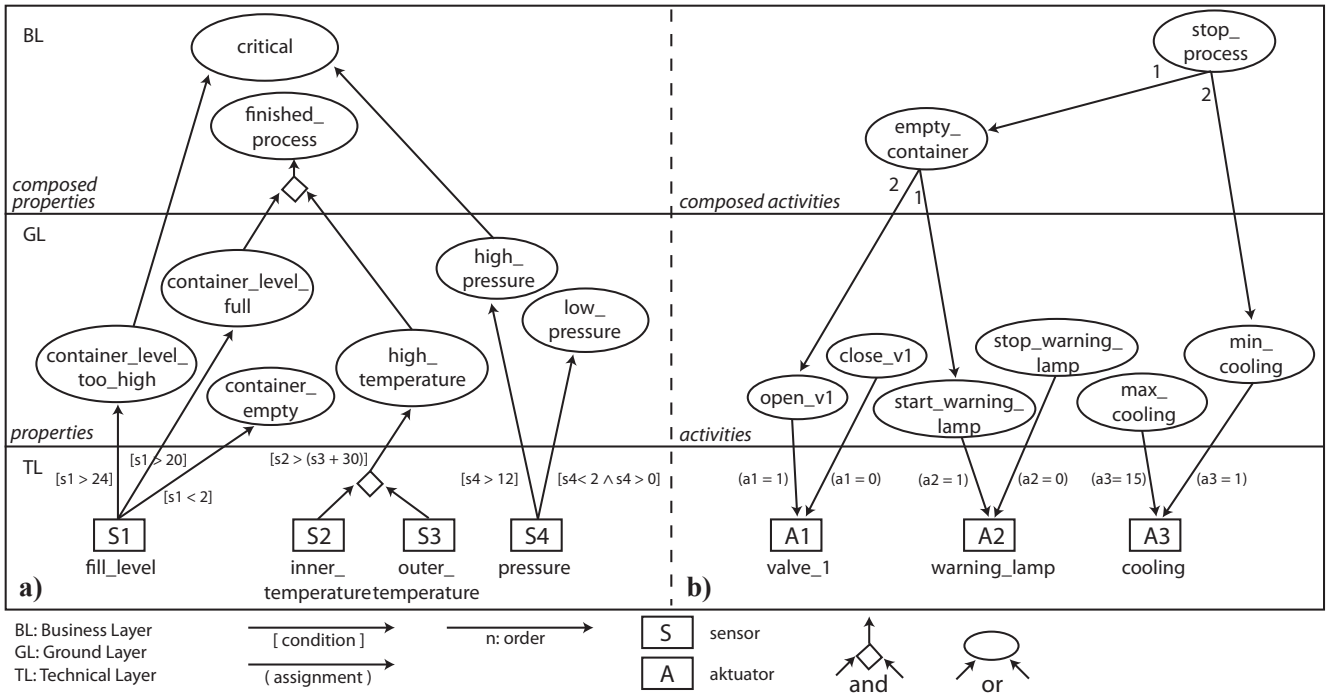

Figure 2. Modeling of properties and activities

critical temperature can be defined which may overlap in the defined range or represent an open interval. A property holds his state until a new state is available so that a discrete sensor may only provide arbitrary changes while a specific property based on this sensor is valid all over life time. While properties directly rely on sensor values, composed properties can be used to summarize multiple properties to a more abstract view. For example multiple properties may lead to a composed property like *critical* which indicates that the system is in a critical state independent from the circumstances that causes this state. Composed properties are boolean functions just like properties and can be used in the last step of behavior modeling to build more abstract and more general rules which significantly simplifies the modeling process. While multiple properties can lead to the validity of an OR-relationship, an AND-relationship is also possible which is represented by the diamond where each of the incoming values need to be valid.

In addition to the sensors depicted in figure 2 sensors can be simulated by software to use further information like time to define properties. For example a *is_holiday* property could be defined which is valid during specific days of the year to define a specific behavior during holidays.

B. Activities

In analogy to the properties, activities are divided into three layers, too. The principle is the same, so that the business layer holds composed activities which are orchestrated by multiple other activities, either composed- or ground activities. In contrast to properties, activities are based on actuators and have a defined order of execution which can be specified by a number on each transition. Activities in the ground layer are used to abstract from hardware and technical details so that they can be used in upper layers without knowledge of the underlying details. In addition, actuators can also be software for example to send emails or a message via SMS. To be able to communicate with actuators and to fulfill the associated task of an activity, assignments can be done from the ground layer to the technical layer which represent the value that shall be assigned to the underlying actuator. An example for the modeling process of activities is depicted in figure 2 b) where one composed activity (*stop process*) initiates multiple further (composed) activities which finally communicate to the hardware. Through the abstraction proposed in the modeling process of properties and activities it is possible to provide basic interactions with the environment like interpreting the sensor values or changing an actuators state by technical engineers while the provided properties and activities can be used by a technical laymen to design an even more abstract view on the environment and provide complex activities.

C. Behavior modeling based on statecharts

Like motivated at the beginning of this section an event-condition-rule may become very complex even for simple use cases like opening a valve when a b specific fuel level is reached. The complexity on the one side arises from the technical parameters that need to be considered

during rule definition and on the other side shows up, when multiple data-values need to be considered to evaluate a rule. The same behavior occurs when the action that shall be triggered by a rule consists of multiple actions that need to be performed like opening multiple valves or manipulating further actuators of a system. To simplify the process of behavior modeling based on rules we use statecharts which use the previously introduced properties and activities to abstract from technical details and to summarize complex states or task. Two examples of statecharts defining rules are depicted in figure 3.

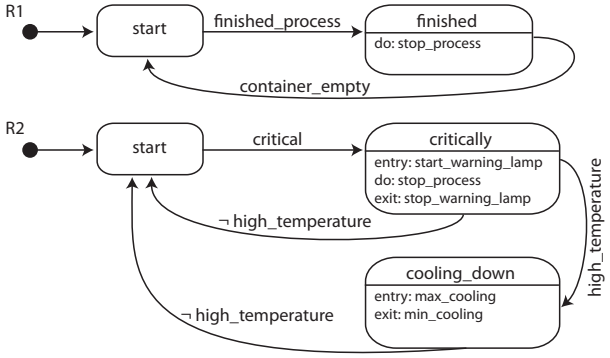

Figure 3. Specifying rules

A rule initially remains in the *start* state which means that the condition to reach another state is not valid. The condition is always based on properties like defined in figure 2 a). Because of the definition of a property as a boolean function, regardless of whether it is an property or a composed property, it can directly be used as an event on transitions because it always evaluates to *true* or *false*. This also allows us to use inverted property values as a condition to specify rules which enter a state on validation of a property and exit that state on the denied property state. Two examples of rules defined by the use of statecharts are depicted in figure 3. The first rule R1 represents the simplest form of a rule with only two states, *start* and *finished*. Properties are used as events at transitions and activities are executed on entering the *finished*-state. The second rule R2 additionally allows to differentiate on the temperature of a sensor after the *critically* state is not valid any more. This way further activities can be executed on the system and more complex rules can be modeled.

The introduced statecharts for behavior modeling allow an very efficient way to define rules for a monitored system. For example the *empty_container* activity triggered in R2 completely abstracts from technical details like which actuators of the system are involved in that step and which parameters need to be set. Through the ordering of the steps during the definition of the abstract activity, the statechart approach even abstracts from execution sequences of necessary steps. Because of the simplicity reached by this procedure the definition of the behavior of a system is not limited to technical engineers but can also be performed by technical laymen. While the technical engineers provide the technical layer and the ground layer of the properties and activities and the technical layman can define the business layer and the rules that shall actually be applied.

D. Integration into act-mobile

The act-mobile system is divided into two parts, the online platform and local gateways. While the online system is used for long-term data storing, online access for monitored systems and statistical analyses, the operation of a monitored system must be guaranteed even if the connection between a gateway and the online system cannot be established. As a result of the necessary reliability and due to performance reasons the defined behavior needs to be applied directly on the gateway without a communication to the online system. On the other hand a gateway often does not have much computing power because it may be realized as a small industrial pc or be integrated into an environment where it needs to keep a low profile. For this reasons a rule engine may be too performance consuming on a gateway which is why we are going to use an model driven approach to produce executable code. Therefore we are going to implement a rule editor for the online system which also allows the definition of properties and activities. Based on these characteristics the rules are designed to complete the task of behavior modeling. Building up on the statecharts, properties and activities we plan to develop a code generator that can be used to generate code that reflects the defined behavior and can simply be executed by a gateway. Through this approach different kinds of gateways can be supported, too.

During the definition of the system behavior some basic and domain independent activities like sending emails are available out-of-the-box and can then be configured. These activities mainly differ from actuator-based activities in that they are executed on the server and so a gateway needs to communicate with the act-mobile online system to trigger these activities. This circumstance will also be considered during the code generation process so that a server communication takes places if necessary. Other properties and activities are usually domain dependent and therefore are implemented depending on the scenario the act-mobile system is used in.

IV. Related Work

The proposed approach addresses two issues in intelligent environments whether they are used in the domain of home automation, industrial monitoring and controlling or any other scenario. The first issue is the domain independence that allows us to use act-mobile for different scenarios. The second issue addresses the modeling of the behavior which can be a very complex task and therefore was limited to

experts that know the monitored environment in detail. A solution for both issues was presented in this paper why we first discuss the architecture of act-mobile and then the process of behavior definition.

A. AmI architecture

While many middleware have been proposed, most of them are domain specific and allow to fulfill a predefined task. Only a few solutions try to address not only a specific domain but to allow the application of a middleware in as many scenarios as possible, while a "one size fits all" solution probably will never exits. The requirements of such a middleware are summarized by Roalter et. al in [14] and basically address the abstraction over heterogeneity, hardware and software interfaces, continuous or discrete data and data types, abstraction over location and context and the development process. To achieve these goals Roalter et. al adapted the ROS middleware which originates from the robotics domain and is an open source robot operating system. While some of the mentioned requirements like the challenges concerning heterogeneous hardware are considered in ROS, the platform does not fit the addressed scenario of act-mobile where a service platform for online access to a monitored facility is one key feature.

Another approach of a common middleware architecture for AmI infrastructures is proposed in the position paper [20] where Bavafa et. al try to address the issues of such an environment by a bottom-up approach. Accordingly, the paper is focused on the easy customization of middleware to different hardware topologies. What they do not consider is the distribution of the whole system through the use of an online service platform and local gateways so that the scenario they are using their architecture does not meet the scenario act-mobile is used in.

B. Behavior modeling

Actually the behavior modeling is a key feature in each intelligent environment whether it is used in the home automation or in industrial facilities. Depending on the domain a resident or a business process expert may be involved in the process of rule definition. Therefore it is necessary to raise the level of abstraction and to provide an easy to handle interface for rule definition. For this purpose Drey et. al proposed a visual and end-user oriented programming interface which is based on a taxonomy that describes the involved environment [21], [22]. Based on this description an editor was implemented which allows to link sensors and actuators through the use of controllers. One key feature of that approach is the development of orchestration rules based on an environment description. While this approach already offers an easy to handle environment for rule definition it lacks a further abstraction of sensors and actuators. The rule definition directly uses the taxonomy which can be compared with the ground layer of the approach proposed in this paper.

Rule based decision making is also used in many other approaches [16], [23] for intelligent environments where the formal structure and other aspects may differ slightly, but they do not abstract from technical details or do not provide an intuitive way for rule definition. Also the use of context information for defining rules is an topic of interest in current research activities [24], [25]. While those approaches explicitly build on ontologies for defining the environment, the proposed approach is also able to consider context information through the different layers used to define properties. As mentioned before, time for example can also be modeled and thereby considered during rule definition. Other context information whether they are based on sensor data (like location) or general information can be used through the use of software in the technical layer simulating an sensor. Ye et. al proposed a top-level ontology [26] that can be used to abstract from technical details and also allows the definition of abstract activities. In contrast to the approach proposed in this paper the top-level ontology consists of information dimensions (like time or any sensor value) and generalizations of that information. The generalized activities are always fired if one of the underlying activities is fired so that only the conditions of the ground activities themselves need to be met. Furthermore, rules are not explicit modeled and a so called application is executed if a connected activity is valid. Applications like settling a phone call can also not be designed hierarchically and thus do not provide a way to design abstract actions.

V. Further work

Different aspects will be considered during future work concerning the proposed act-mobile platform and the behavior modeling. Regarding the act-mobile architecture especially the gateway design and structure will be part of further improvements. Until now we just consider the use case where one gateway is integrated into one facility that is monitored. This scenario is very basic and does not meet all real word applications where for example one central gateway could be installed to monitor and control a huge building while multiple small gateways are connected to the central one and each one individually manages one floor of the building. This style of architecture rises new challenges due to the distribution of gateways and the connected sensors and actuators as well as the evaluation of rules. A concept for a joint fact-basis or an gateway-comprehensive rule execution needs to be developed.

Another aspect of the future work focuses on context information. While we only mentioned that the use of context information is possible through the technical layer and the definition of properties, in future work we are going to address the concepts of context recognition and context based behavior in depth where one possible approach may

be the use of ontologies to describe an environment. This step also includes the detection of inconsistencies as well as the handling of uncertainty in context so that the additional knowledge that is introduced by the use of ontologies is used efficiently. Furthermore we are going to introduce a formal definition of the proposed concepts. Based on these definitions it shall be possible to formally describe the behavior of the system and to apply verification methods to it.

One more concern that will be addressed in future work is the security and privacy management in the act-mobile system. Existing solutions implemented in act-mobile do not exceed the use of standard technologies like SSL for communication. But especially with the expansion of the system and the system architecture new security challenges need to be considered.

VI. Conclusion

In this paper we proposed an architecture for pervasive computing environments which application is not limited a specific domain. The message oriented architecture implements the publisher/subscribe pattern through the use of message queues. Besides the use of message queues the extendability and domain-independence is reached through a component architecture which allows an easy integration of new functionality through components. Another distinction to many other approaches is the integration of a centralized service platform which can be accessed online while local gateways are directly connected to sensors and actuators and establish the communication to the service platform. Through this architectural style and the possibility to access all data of a monitored system through the internet with different client systems, the act-mobile system is well suited for the surveillance and control of industrial facilities as well as for buildings and other private facilites. We further introduced a concept for behavior modeling based on properties and activities which can be connected through state charts. Both, properties and activities, are used to abstract from technical details and can be orchestrated to abstract views of the system. Through the use of abstract views the process of behavior modeling is simplified and rules can be defined without technical knowledge. The proposed approach also allows the use of model driven technics to generate code that can be executed on the gateways to realize the designed behavior.

References

[1] N. Georgantas, S. B. Mokhtar, Y.-D. Bromberg, V. Issarny, J. Kalaoja, J. Kantarovitch, A. Gerodolle, and R. Mevissen, "The amigo service architecture for the open networked home environment," in *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 295–296.

[2] . Fides-Valero, M. Freddi, F. Furfari, and M.-R. Tazari, "The persona framework for supporting context-awareness in open distributed systems," in *Ambient Intelligence*, ser. Lecture Notes in Computer Science, E. Aarts, J. Crowley, B. de Ruyter, H. Gerhuser, A. Pflaum, J. Schmidt, and R. Wichert, Eds. Springer Berlin / Heidelberg, 2008, vol. 5355, pp. 91–108.

[3] A. Gárate, N. Herrasti, and A. López, "Genio: an ambient intelligence application in home automation and entertainment environment," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, ser. sOc-EUSAI '05. New York, NY, USA: ACM, 2005, pp. 241–245.

[4] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *J. Netw. Comput. Appl.*, vol. 28, pp. 1–18, January 2005.

[5] D. phys Ilias Sachpazidis, "@home: a modular telemedicine system," in *Proc. 2 nd workshop on Mobile Computing in Medicine*, 2002, pp. 87–95.

[6] L. M. Camarinha-matos and H. Afsarmanesh, "Telecare: Collaborative virtual elderly care support communities, in the," *Journal on Information Technology in Healthcare*, vol. 2, pp. 73–86, 2004.

[7] R. A. Brooks, "The intelligent room project," in *Proceedings of the 2nd International Conference on Cognitive Technology (CT '97)*, ser. CT '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 271–.

[8] A. Waibel, T. Schultz, M. Bett, M. Denecke, R. Malkin, I. Rogina, R. Stiefelhagen, and J. Yang, "Smart: The smart meeting room task at isl," in *in Acoustics, Speech, and Signal Processing (ICASSP '03). 2003: IEEE*, 2003, pp. 752–755.

[9] J. C. Augusto and C. D. Nugent, "The use of temporal reasoning and management of complex events in smart homes." in *ECAI'04*, 2004, pp. 778–782.

[10] G. Acampora, V. Loia, M. Nappi, and S. Ricciardi, "Ambient intelligence framework for context aware adaptive applications," *Computer Architectures for Machine Perception, International Workshop on*, vol. 0, pp. 327–332, 2005.

[11] A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis, "A survey of semantics-based approaches for context reasoning in ambient intelligence," in *Constructing Ambient Intelligence*, ser. Communications in Computer and Information Science, M. Mhlhuser, A. Ferscha, and E. Aitenbichler, Eds. Springer Berlin Heidelberg, 2008, vol. 11, pp. 14–23.

[12] E. Christopoulou, C. Goumopoulos, and A. Kameas, "An ontology-based context management and reasoning process for ubicomp applications," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, ser. sOc-EUSAI '05. New York, NY, USA: ACM, 2005, pp. 265–270.

[13] N. Georgantas, V. Issarny, S. B. Mokhtar, Y.-D. Bromberg, S. Bianco, G. Thomson, P.-G. Raverdy, A. Urbieta, and R. S. Cardoso, "Middleware architecture for ambient intelligence in the networked home," in *Handbook of Ambient Intelligence and Smart Environments*, H. Nakashima, H. Aghajan, and J. C. Augusto, Eds. Springer US, 2010, pp. 1139–1169.

[14] L. Roalter, M. Kranz, and A. Möller, "A middleware for intelligent environments and the internet of things," in *Proceedings of the 7th international conference on Ubiquitous intelligence and computing*, ser. UIC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 267–281.

[15] J. C. Augusto, "Ambient intelligence: the confluence of ubiquitous/pervasive computing and," *Artificial Intelligence*, pp. 213–234, 2007.

[16] R. Etter, P. D. Costa, and T. Broens, "A rule-based approach towards context-aware user notification services," in *Proceedings of the 2006 ACS/IEEE International Conference on Pervasive Services*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 281–284.

[17] H. Hagras, F. Doctor, V. Callaghan, and A. Lopez, "An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 1, pp. 41–55, 2007.

[18] G. Riva, F. Vatalaro, F. Davide, and M. Alcaiz, "A flexible architecture for ambient intelligence systems supporting adaptive multimodal interaction with users," pp. 97–120, 2005.

[19] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, pp. 231–274, June 1987.

[20] M. Bavafa and N. Navidi, "Towards a reference middleware architecture for ambient intelligence systems," in *Knowledge Engineering, 2010 8th International Conference on ICT and*, nov. 2010, pp. 98 –102.

[21] Z. Drey and C. Consel, "A visual, open-ended approach to prototyping ubiquitous computing applications," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, 2010, pp. 817 –819.

[22] Z. Drey, J. Mercadal, and C. Consel, "A taxonomy-driven approach to visually prototyping pervasive computing applications," in *Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages*, ser. DSL '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 78–99.

[23] F. Paganelli and D. Giuli, "An ontology-based context model for home health monitoring and alerting in chronic patient care networks," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*, ser. AINAW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 838–845.

[24] I. Cafezeiro, J. Viterbo, A. Rademaker, E. H. Haeusler, and M. Endler, "A formal framework for modeling context-aware behavior in ubiquitous computing," in *Leveraging Applications of Formal Methods, Verification and Validation*, ser. Communications in Computer and Information Science, T. Margaria and B. Steffen, Eds. Springer Berlin Heidelberg, 2009, vol. 17, pp. 519–533.

[25] A. Ranganathan and R. H. Campbell, "A middleware for context-aware agents in ubiquitous computing environments," in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, ser. Middleware '03. New York, NY, USA: Springer-Verlag New York, Inc., 2003, pp. 143–161.

[26] J. Ye, G. Stevenson, and S. Dobson, "A top-level ontology for smart environments," *Pervasive and Mobile Computing*, vol. 7, pp. 359–378, June 2011.