The eHomeConfigurator Tool Suite

Ulrich Norbisrath, Christof Mosler, and Ibrahim Armac

Department of Computer Science 3, RWTH Aachen University,
Ahornstr. 55, 52074 Aachen,Germany,
{uno|christof|armac}@i3.informatik.rwth-aachen.de

Abstract. New developments and decreasing costs of electronic appliances enable the realization of pervasive systems in our daily environment. In our work, we focus on eHome systems. The price of individual development and adaption of the software making up these systems is one of the major problems preventing their large-scale adoption. In this paper, we introduce an approach built upon functionality composition for automatic service configuration in different environments. We transform the repetitive development process to a single development process followed by a repetitive configuration process. This configuration process is supported by our tool suite, the eHomeConfigurator. The result is a configuration graph, capable of describing dependencies and contexts of components in the eHome field. The tool suite is used to configure and deploy various services on different home environments. Compared to the classical development process, the effort for setting up eHome systems is reduced significantly.

Proceedings of the 1st International Workshop on Pervasive Systems (PerSys)
Montpellier, France, LNCS 4278, p. 1315-1324,
ISBN 3-540-48273-3, Springer, 2006 (copyright)

1 Introduction

Appliances usable in pervasive computing environments, such as computer controlled lamps, cameras, or various sensors, are already affordable for most home-owners. Hence, from this point of view, the hardware for the realization of smart home environments is already available at reasonable costs. This is proven by various setups of such smart home environments. We call those environments eHomes or eHome systems [3,9,1]. All these implementations are either research or hobby projects. The question arising is: Why are eHomes not more wide spread? One of the main barriers blocking this is the price of such systems. Even if appliances are affordable, the software driving the eHome is rather expensive since it is mostly developed or adapted for every single eHome. A complete software development process per case is not affordable for everyone.

As software engineers, we are particularly interested in simplifying the software development process, arising when implementing a service for a specific home environment. Our vision: If the software for eHomes could be reused, and its adaptation and configuration could be automated, one of the price barriers for mass-market home automation would be broken. The key point of our research is the reduction of the effort for software development and adaption.

The classic development process for eHome systems is carried out in full for each new eHome environment. In our process, the development has only to be done once for all regarded environments. The repetitive portion is reduced to the level of mere specification of the environment and the services, as well as interactive configuration of the given service components into the eHome system with no coding overhead. Our goal is the support of composition and configuration of services for eHome systems. In this context, we introduce a special specification, configuration, and deployment process. We will refer to this as the eHome-SCD-process. This process is supported by our tool suite, the eHomeConfigurator, which will be presented in this paper.

2 Scenario

We assume the following scenario will be typical for establishing future eHome environments: First, an inhabitant has the wish to install an eHome service in his/her home environment. He/she is the *customer*. The customer will specify which functionality should be provided by his/her eHome system. He/she will do this by contacting a *provider* or using a tool which contacts the provider. The provider offers eHome hardware, software, and configuration support. In our case, the customer will consult a configuration tool delivered by the provider for selecting a service offering the desired functionality. The services are described and classified on an abstract level to make an intuitive selection possible. The tool will provide suggestions for appliances needed to enable the selected functionality and will take the environment's existing infrastructure into account. After the installation of the proposed appliances, the software will support the configuration process of the software and deploy it automatically.

We distinguish between basic and integrating services offered by components. *Integrating services* are services composed of various basic services which provide interfacing with single appliances. These integrating services usually represent the functionality the inhabitants are interested in.

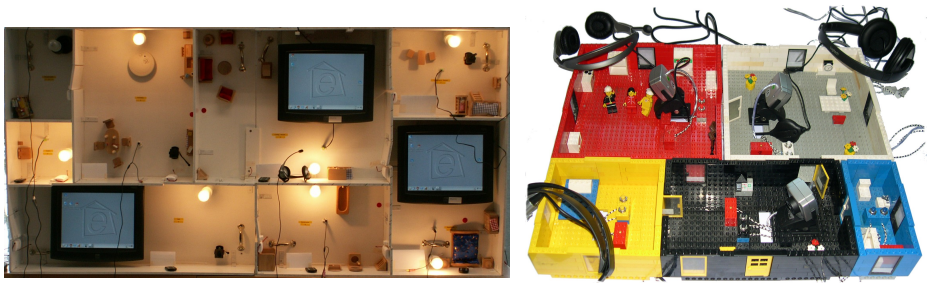

Fig. 1. eHome demonstrators

In our evaluation, we offer the following integrating eHome services: **LightingService:** This service offers simple functionality to turn on and off the illumination in

specific areas in the home environment via manual switches or buttons. **LightMotionService:** This service offers the possibility to turn on and off the illumination via motion detection. **SecurityService:** If activated, the security service offers surveillance of a home via detecting intrusion and raising accordingly the alarm and informing an inhabitant and/or the police. **MusicFollowsPersonService:** Combining electronically controlled multimedia devices and person detection, this service lets a music stream associated with a specific person follow this person through his/her home. If one inhabitant switches on the music in the living room and goes to the kitchen, the music stops playing in the living room and continues in the kitchen. **AllOnService:** This service switches on all the lights in the whole house. For example: If an inhabitant is awoken by a noise in the middle of the night, he/she can switch everything on. **AllOffService:** This service ensures that all electronic appliances are deactivated, such as in the case, when all inhabitants have left the house.

We have tested our scenario in various environments: We have built two miniature demonstrators (see figure 1). The first one, on the left, (presented on a workshop in Tartu, Estonia [12]) is built from wood and equipped with available eHome appliances (X10 lamps, movement sensors, switch panels, speakers, and webcams [18]). Whereas for the second one (presented at UbiComp2005 [11]), we have used Lego and equipped it with self-built Lego lamps, switches, USB-webcams, and USB-audio systems. Our third test environment is a real house of our cooperation partner located in Duisburg (see section 5). It is equipped with European Installation Bus, Honeywell, and RFID sensors. This diversity of environments and technology standards used, forms a realistic scenario for evaluating the described eHome process. We use these results for proving and improving our concepts realized in the eHomeConfigurator tool suite [4].

3 Process

Current eHome systems are developed individually for each customer by a classic development process. The classic development process includes for every new environment the following steps: **Requirements Engineering:** The customer has to decide beforehand which services he/she wants in his/her house. **eHome system development:** The concrete realization in terms of appliances, communication infrastructure, and middleware must be worked out together with specialists from hard- and software providers. The implementation of the eHome system is the next step. Usually this will be a hand-coded and very environment specific solution. **Execution and Maintenance:** The software is installed and deployed by the provider. Later changes and extensions demand further coding and development time, as described in the previous step. **Deinstallation:** The software components are de-installed, and the contact between the customer and the provider ends.

3.1 The SCD-process

The first step to the reduction of repetitive development is to shift the development away from the per case related development to an a-priori development of reusable components. What remains is a repetitive configuration process consisting of specification,

configuration, and deployment. Still, the configuration is a challenging task when done manually. By introducing functionality composition, allowing automatic resolution of subservices, and corresponding appliances for given integrating services, the configuration process can be partly automated. We call this supported configuration process the SCD-process.

The idea of the SCD-process is to establish an iterative chain of procedural techniques to automate the creation of the eHome system as much as possible. This means that we are looking for ways to automate and support the process of specifying, configuring, and deploying an eHome system into the normal home – transforming the regular home into an eHome. This is achieved by means of tool support, reuse, and mere configuration of software components for providing eHome services. It is essential that the shift from a normal home to an eHome does not involve developing the software for that home but merely configuring and deploying it. As the result, the specification of the home environment, selection of the desired eHome services, and installation of necessary appliances are the only activities performed manually. Current configuration management research mainly deals with manual configuration management and software deployment [17,14]. Here, the configuration and deployment of the given components must also be done automatically and support functionality composition. Hence, we are focusing not on versioning aspects but rather on semantic support. The three phases of the SCD-process are described in the following:

Specification of the eHome environment and required services During this phase, the architectural information about the eHome is captured – how the rooms in the home are located and connected with the different location elements such as doors and windows. The existing appliances and their location in the home environment are described. The already existing eHome services are also identified – when modifying the configuration of the eHome, the already existing eHome services have to be specified. Whenever new eHome services are needed, they are selected and added to the specification. Only integrating services are selected. Along with the eHome environment, the services used later in the eHome environment, plus the required appliances and functionalities, need to be defined and specified beforehand, as well.

Automatic configuration of the selected services The services selected in the specification phase are automatically configured. This means that necessary appliances that are still missing in the eHome are added to the configuration, with the impact that the customer will have to buy them before deploying. Likewise, required sub-services that are missing in the specification are selected to meet the functional requirements of the selected services. For each location with active services, corresponding service objects are allocated. For example, if the lighting service needs at least one lamp per room and one switch to control the lamp, these appliances are added to the configuration. Furthermore, the corresponding driver component services for the lamp and switch controllers are added to the configuration. After finishing this traversal, the configuration graph includes all necessary information for the deployment.

Deployment of the service configuration onto the service gateway in the eHome The software components specified and configured during the first two phases are deployed automatically onto the service gateway residing in the eHome. The software components are also initialized properly and launched automatically. The third phase of the eHome-SCD-process concerns the deployment of the eHome configuration graph. Up to now, the deployer tool is only realized for the OSGi framework [5]. The implemented algorithm is very straight forward and should be easily transferable to other middlewares. The algorithm consists of five steps. First, the software components (in OSGi called bundles) are loaded. Second, the components are stated. Third, the references between the runtime model and the runtime components are built. Fourth, the software component are initialized in correspondence to their dependencies. And fifth, the service object interfaces in the software components corresponding to the integrating services are executed. Possible extensions of this deployment approach concern dynamic reconfiguration and the integration of different middlewares.

4 Tool support

Our tool suite, the eHomeConfigurator, supports the eHome-SCD-process. In this subsection we describe how this tool can be used in practice.

Before the customer can use the tool suite, the provider or the software component developer has to provide a set of software components (in OSGi called bundles) and their service descriptions in a semantic context. To specify a service, all its semantic requirements and exports must be given. Its URL has to be given to locate the compiled component code. Furthermore, the service must be classified to be a top level (selectable by the eHome customer) service or not. An integrating service will be selectable by the customer. In figure 2, a security service is specified. It requires the functionalities for intrusion detection, alarm raising, and the alarm activation state. Some services, especially driver services, can control appliances. They also have to be specified including their describing attributes, such as address numbers for X10-lamps.

After launching the eHomeConfigurator at the customer's home, his/her environment has to be modeled. In our case (see the Lego demonstrator in figure 1) just kitchen, living room, bedroom, hall, and bathroom and their connections have to be modeled (see figure 3). Until now, no appliances have been specified; there are no software controllable appliances in the customer's environment. This means that all required appliances to fulfill the selected services will be recommended by the tool suite.

In the next step, the inhabitant has to select which services should be installed in his house (see figure 4). Each service will be assigned to desired locations. Already installed appliances can be taken into account while selecting the locations to fulfill the respective service. As in our example no appliance are installed, this step is not relevant. In our case, the inhabitant selects the first five services from the service list and configures the services in all rooms (he selects `Select All Locations`) and configures the services with the minimal set of needed appliances (he/she selects `Necessary Devices Only`).

In the next step, the inhabitant can fine tune the room-selection: Some services such as **AllOnService**, which turns on all the lights in the complete environment with one

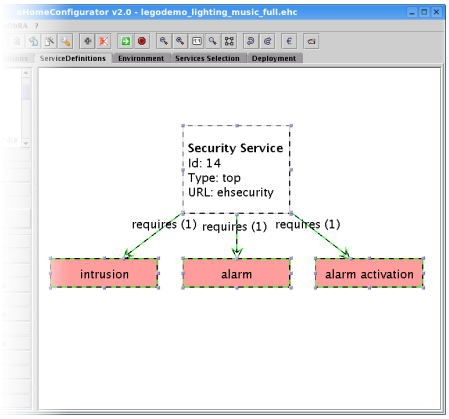

Fig. 2. eHomeConfigurator: service specification

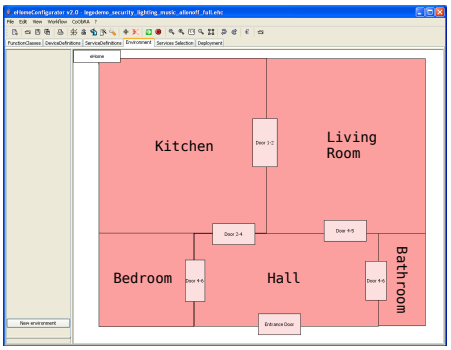

Fig. 3. eHomeConfigurator: empty environment

button, might only be accessed from special locations, such as the bedroom or the hall. They need not be accessible from every room.

After providing this information, the configuration tool can compute which software realizations of the selected top level services exist for the given parameters and environment. In our scenario, the inhabitant is offered the realization using drivers for the Lego demonstrator as well as alternatives for other drivers, resulting in other required hardware and software components, such as hardware used in another demonstrator. In the depicted scenario, also an X10-based realization [18] in one or all rooms is possible. The system notices that it can realize the illumination-functionality via an X10-controller service or a Lego Lamp Controller service (see figure 6).

When this selection is done, the configuration tool finishes the computation of necessary software components and appliances. The inhabitants can now buy the appli-

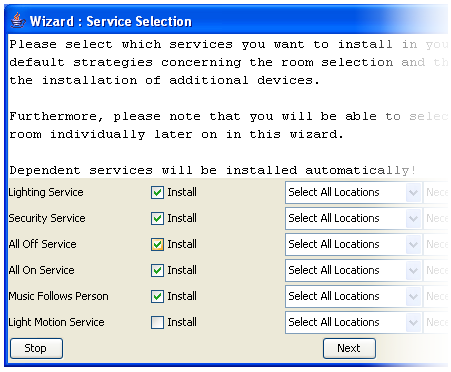

Fig. 4. eHomeConfigurator: service selection

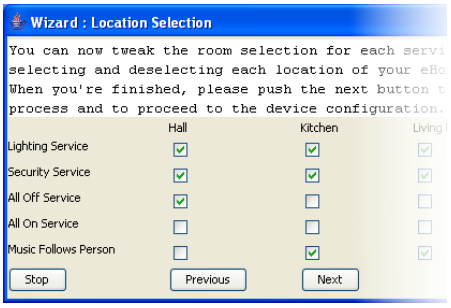

Fig. 5. eHomeConfigurator: location selection

Fig. 6. eHomeConfigurator: alternatives of software components

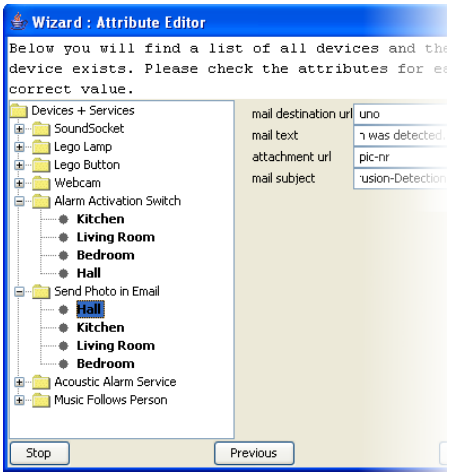

Fig. 7. eHomeConfigurator: attribute editor

ances and enter their hardware addresses and other parameters in our tool (see figure 7). These parameters could be the address numbers of the Lego lamps (i.e. 2 for the kitchen) or email addresses (i.e. uno@i3.informatik.rwth-aachen.de), and message texts (i.e. "intrusion detection in the kitchen") for the security service. In the figure, the parameters for the notification service of the security service are depicted.

As a first result, all appliances required to fulfill the selected services have been added to the environment (see figure 8). These have to be installed in the corresponding room before performing the actual deployment.

A further result is the deployable configuration graph, which can be brought to life by our deployment tool. This graph is not intended to be viewed or processed by the customer, but is used for the automatic deployment and as a visual entry point for debugging. In the context of our scenario, this graph contains nodes representing the service objects, appliances, and their attributes, associated with the different locations. In figure 9, we see a section of the configuration graph showing the **MusicFollowsPerson** service in the living room. It shows only the graph nodes related to the living room and the **MusicFollowsPerson** service. These nodes represent appliances (Sound Device, On/Off Switch, and Webcam), subservices (Person Detector, Motion Controller, and Switch), attributes (Destination, Camera Number), and states (`switch` and `person`). Furthermore, the graph edges represent the dependencies of software components (the Music Follows Person - object uses the SoundRouter - object), physical contain relations (the Sound Device is in the Living-room), responsibilities of runtime software components for environment elements (the Soundrouter - object is responsible for routing sound to the Living-room), and links to attributes (attr edges) and states (state edges).

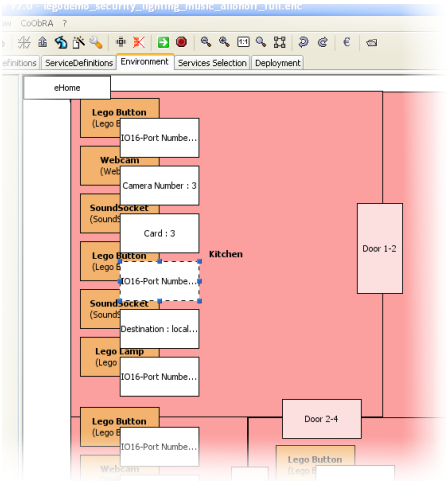

Fig. 8. eHomeConfigurator: environment with appliances

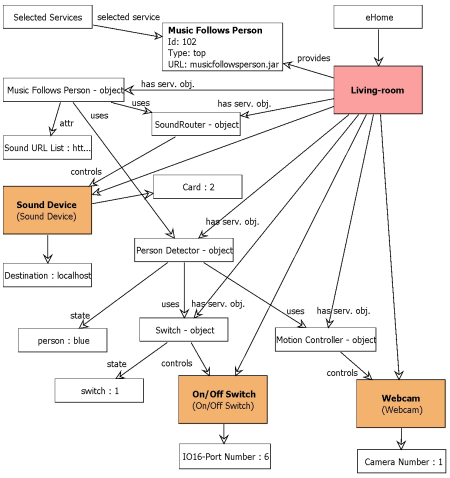

Fig. 9. Deployable configuration graph for the MusicFollowsPerson Service

5 Related Work

While a lot of research is being done in the field of ubiquitous and pervasive computing, the main focus is usually on the development and study of new types of applications. However, there are few related scientific projects which also have the simplification of the development process in the center of their interest. The following section aims at pointing out similarities and differences between the previously presented eHomeConfigurator and some of those projects.

A number of prototype eHome buildings are located in different parts of Germany. In the city of Duisburg, a building equipped with state of the art technology has been constructed by inHaus in close cooperation with The Fraunhofer Institute for Microelectronic Circuits and Systems (IMS) for the purpose of developing products and strategies in that segment and being able to perform tests in a real environment [3]. Similar to this building, a different project has been realized by T-Com and WeberHaus in Berlin, Germany. In contrast to the building in Duisburg, this eHome was built specifically using technology which is either already available or will be available within the next few months [9] (similar to our wooden demonstrator). There is one common character trait for each of these eHomes. All of them have been developed and built individually, using custom software components which had to be realized through a vast amount of work being done by expensive experts. As a direct result, large investments have been necessary in order to realize those buildings.

SmartHome User Interface [15] has been developed by SIKT, HK\R and EnerSearch in Sweden. This tool provides a user interface to view and manipulate appliances over the internet. It is also capable of displaying the environment three-dimensionally to simplify access to those appliances. In contrast to the tool presented in this paper, *SmartHome User Interface* does not allow the user to model complex services or to

configure them in an environment. Therefore, it provides no support during the eHome configuration process.

ECT [2] is a toolkit to support developing and deploying installations for ubiquitous computing. Its main focus is the interaction of heterogeneous soft- and hardware components. It establishes three distinct layers for different types of users. The lowest layer represents the core implementation of ECT which is only of interest for a certain kind of experienced developers. On a second layer, other programmers can develop new components in accordance to a component approach similar to Java Beans. On the highest level, unexperienced users can make use of the graphical user interface to assemble these components.
In contrast to the approach presented in this paper, this toolkit provides no means to model the environment in which the ubiquitious computing takes place. It only focuses on the creation of new services through component interaction and on how to make this process accessible to unexperienced users. When using the eHomeConfigurator, service modeling is not done by the end user, but by experts. End users only choose from existing services and configure them according to their taste. Consequently, the modeled services can be applied to an arbitrary number of environments, which is not the case for services created with ECT. A similar approach is the *CAMP* [10] project. It does not implement different abstraction layers, but rather focuses on the end user level. Extensive research on how end users with no technical experience perceive applications has been conducted.

The *Open Services Gateway Initiative* (OSGi) [6] specifies a set of software application interfaces (APIs) for building open-service gateways. Central to the OSGi specification [5] is the *service framework* with a service registry. Components register their services at the service registry where other applications may retrieve and use them. Based on the concept of residential gateways [16,8], the open services gateway specification describes an approach that permits coexistence of and integration with multiple network and device access technologies. In addition, components may be added implementing new technologies as they emerge. Interaction is enabled via Java interfaces, without relying on proxy objects. While other systems (e.g Jini) are decentralized, the OSGi approach is a centralized system, which simplifies system maintenance to the disadvantage of distribution aspects. OSGi is the middleware in our project.

6 Conclusion & Outlook

Automatic configuration of services and appliances is an important requirement for the acceptance of eHome systems. In this paper, we presented an approach for modeling specific eHome environments, comprising different services and appliances and performing automatic configuration of various eHome services corresponding to this environment. We have developed a tool suite called the eHomeConfigurator to support all steps of our SCD-process: specification, configuration, and deployment of eHome systems. The eHomeConfigurator calculates and proposes different combinations of appliances and drivers. The resulting configuration graph contains all the information needed for automatic deployment.

In the classical development process, where intense requirement analysis, manual specification and formalization, and functionality and glue programming is required, we need multiple specialists for performing these tasks for every new environment. Our approach offers a convenient way to shift the need of specialists to the a-priori development phase specifying functionalities, appliances, and coding services. The remaining tasks can be carried out by most customers. Of course, aspects concerning the user interface can still be improved in future work; localization of appliances and detection of their attributes (for example with UPnP [13]) is ongoing work in the field of pervasive computing. Such extensions should be easily includable in our tool suite.

Other future work related to our project concerns the improvements of the MDE-based software development process of our tool suite itself and the enhancement of the functionality description with more fine-grained parametric contracts [7].

References

1. Beisheim Holding GmbH. FutureLife – Das Haus der Zukunft. `http://www.futurelife.ch/` (21.08.2006), 2005.
2. Humble J. Greenhalgh C., Izadi S., Mathrick J. and I. Taylor. ECT: A toolkit to support rapid construction of ubicomp environments. In *In Proceedings of the Sixth International Conference on Ubiquitous Computing (UBICOMP'04)*, September 2004.
3. inHaus Duisburg. Innovationszentrum Intelligentes Haus Duisburg. `http://www.inhaus-duisburg.de` (21.08.2006), 2005.
4. Ulrich Norbisrath, Priit Salumaa, and Adam Malik. eHomeConfigurator. `http://sourceforge.net/projects/ehomeconfig`, 2006.
5. Open Services Gateway Initiative. OSGi Service Platform Specification. `http://www.osgi.org/osgi_technology/download_specs.asp` (2.3.2005), 2002.
6. Open Services Gateway Initiative Alliance. OSGi Service Platform. `http://www.osgi.org/osgi_technology/download_specs.asp#Release_3` (2.3.2005), April 2003. Release 3.
7. Ralf Reussner, Jens Happe, and Annegret Habel. Modelling parametric contracts and the state space of composite components by graph grammars. In *FASE*, pages 80–95. Springer-Verlag, 2005.
8. Takeshi Saito, Ichiro Tomoda, Yoshiaki Takabatake, Junko Ami, and Keiichi Teramoto. Home Gateway Architecture and Its Implementation. *IEEE Transactions on Consumer Electronics*, 46(4):1161–1166, November 2000.
9. T-Systems. Wohnen in der digitalen welt. *Best Practice - Ausgabe für den Mittelstand*, pages 17–18, March 2005.
10. Khai N. Truong, Elaine M. Huang, and Gregory D. Abowd. CAMP: A magnetic poetry interface for end-user programming of capture applications for the home. In *Ubicomp*, pages 143–160, 2004.
11. Ulrich Norbisrath, Priit Salumaa, Adam Malik. eHome Specification, Configuration, and Deployment. `http://ubicomp.org/ubicomp2005/programs/demos.shtml`, 2005. Demonstration Paper D15 on UbiComp2005.
12. Ulrich Norbisrath, Priit Salumaa, Adam Malik. Specification, Configuration, and Deployment in eHome Systems. `http://math.ut.ee/~peeter_l/seminar/eelmised/05k/ehome.html`, 2005.
13. UPnP-Forum. UPnP-Forum Homepage. `http://www.upnp.org`, 2005.

14. André van der Hoek. Integrating Configuration Management and Software Deployment. In *Proceedings of the Working Conference on Complex and Dynamic Systems Architecture (CDSA 2001)*, 2001.
15. Rolf Raven Elmar van Dijk and Fredrik Ygge. Smarthome user interface: Controlling your home through the internet. *ISES*, 8, 1996.
16. Dave L. Waring, Kenneth J. Kerpez, and Steven G. Ungar. A newly emerging customer premises paradigm for delivery of network-based services. *Computer Networks*, 31(4):411–424, February 1999.
17. Bernhard Westfechtel and Reidar Conradi. Version Models for Software Configuration Management. *ACM Computing Surveys*, 30(2), June 1998.
18. X10. Protocol Specification. `http://www.marmitek.com/en/basisimages/x10_protocol.PDF` (30.5.2005), 2004.