Tool Support for the eHome Specification, Configuration, and Deployment Process

Ulrich Norbisrath
Christof Mosler
Department of Computer Science 3, RWTH Aachen University,
Ahornstr. 55, 52074 Aachen, Germany,
{uno|christof}@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. eHomes are homes equipped with electronic devices and a software system offering services incorporating these devices and external services. The configuration of software components making up these systems is one of the major problems preventing their public use. In this paper, we describe the eHome domain related process of specification, configuration and deployment, and the positive impact of tool support on this process. We present our tool suite, the eHomeConfigurator, supporting the configuration and deployment of various services on different home environments.

1 Introduction

Appliances usable in pervasive computing environments such as computer controlled lamps, cameras, or various sensors are already affordable for regular 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 [inH05, TS05, Bei05, Phi06]. We call those environments eHomes or eHome systems. All these implementations are either research or hobby projects. The question arising is: Why are eHomes not more conventional? 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.

To reduce these development costs and allow a more interactive setup of an eHome system, we must identify the reusable parts and parametrize changing parts of the software system. Our vision is: If the adaptation and configuration of software for eHome systems could be automated, one of the price barriers on home automation mass-market would be broken. By introducing specification support and semantic-based software composition, we shift the classic development process toward a customer-oriented specification, configuration, and deployment process. We will refer to this as the eHome-SCD-process. From the view point of the customer, the process is reduced to the level of mere specification of the environment and service selection, and interactive configuration of the given service

components into the eHome system. From the view point of the service provider, only appliance drivers and environment-independent service components must be implemented as well as their service descriptions.

This paper is structured as follows: in the following section, different services and simulation environments are described. In section 3, we discuss the three phases of our SCD-process. Based on this process, we describe the structure of the supporting eHome-Configurator tool suite in section 4. Section 5 shows the application of the tool suite in practice. In section 6, we will discuss related approaches and frameworks supporting automatic configuration in component-based systems, especially eHome systems. Finally, we give a conclusion of this paper and an outlook.

2 Scenario

We assume the following process will be typical for establishing future eHome environments: First, an inhabitant will specify which functionality should be provided by his/her eHome system. The inhabitant will consult a configuration tool for selecting integrating services offering the desired functionality. They are described and classified on an abstract level to make an intuitive selection possible. Such services can involve lighting, security, or multimedia functionalities. 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.

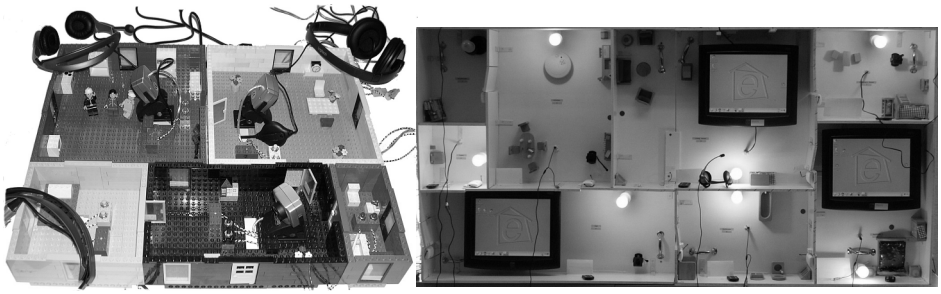

Abbildung 1: eHome demonstrators

In our evaluation, we offer the following top-level 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 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, in the case, when all inhabitants have left the house.

We have simulated this process in various environments: We have built two miniature demonstrators (see figure 1). The first one (presented on a workshop in Tartu [Ulr05b]) is built from wood and equipped with available eHome appliances (X10 lamps, movement sensors, switch panels, speakers, and webcams [X1004]). Whereas for the second one (presented at UbiComp2005 [Ulr05a]) we have used lego and equipped it with 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 6). 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 our eHome process. We use these results for proving and improving our concepts realized in the eHomeConfigurator tool suite [NSM06].

3 eHome development process

Current eHome systems are developed individually for each customer. The customer has to decide beforehand which services he/she wants in his/her house. 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. Later changes and extensions demand further coding and development time.

The idea of the SCD-process is to establish an iterative chain of procedural techniques to automate the creation of the eHome 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 the normal home to an eHome does not involve developing the software but merely configuring and deploying

it. Furthermore, the configuration and deployment of the given components must also be done automatically. 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.

3.1 Specification of the eHome environment and necessary 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 given appliances and their location in the home environment are described – in which rooms or on which location elements the devices are positioned. 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 top-level 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.

3.2 Automatic configuration of the selected services

The services selected in the specification phase are automatically configured. This means that the necessary appliances still missing are added to the configuration with the impact that the customer will have to buy them before deploying. Likewise, the required sub-services that are missing 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.

3.3 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 [Ope02].

The implemented algorithm is very straight forward and should be easily transferable to other middlewares.

The algorithm consists of five steps. First, load of the software components (in OSGi called bundles). Second, start of the components. Third, built of the references between the runtime model and the runtime components. Forth, software component initialization in correspondence to their dependencies. And fifth, execution of the service object interface in the software components corresponding to the integrating services.

Possible extensions of this deployment approach concern dynamic reconfiguration and the integration of different middlewares.

4 eHomeConfigurator tool suite

Figure 2 illustrates the structure of the eHomeConfigurator tool suite. The eHomeConfigurator tool suite has four main tools. The three: Specificator, Auto-Configurator, and Deployer support each directly the respective SCD-process phase. The fourth tool, the DataHolder, is responsible for the encapsulation of the eHome model instance and is used by all of the other three tools.

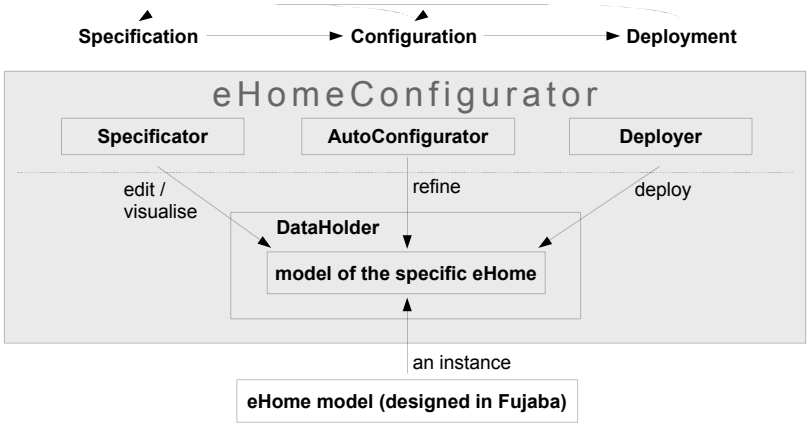

Abbildung 2: The structure of the eHomeConfigurator tool suite

In general, the aim of the eHomeConfigurator tool suite is to support the SCD-process, i.e. to provide to end-users an interface to the eHome model instance. The tool suite is used to carry out the changes on the eHome model instance throughout the entire SCD-process. The tool is also responsible for the persistency of the model instance.

The eHomeConfigurator tool suite has been developed in the way that it can be run as a standalone application. This enables the work on the eHome model instance separately from the eHome system. The eHomeConfigurator can also be started as a component bundle in the OSGi framework. This allows changing and monitoring the eHome model

instance during the runtime of the eHome system, making the eHomeConfigurator tool suite an integral part of this system.

5 eHomeConfigurator in practice

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 service or not. A top level service will be selectable by the customer. Top level services usually do not export functionalities for higher services. In figure 3, a security service is specified. It requires the functionalities for intrusion detection, alarm raising, and alarm activation state. Some services, especially driver services, can control appliances. They also have to be specified including their describing attributes, like address numbers for X10-lamps.

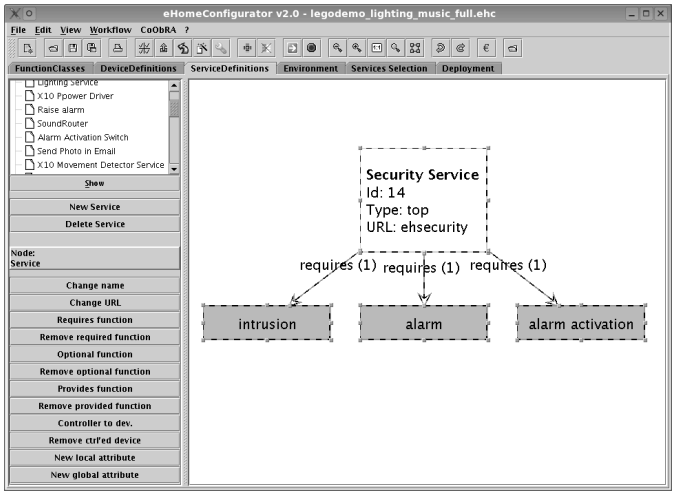

Abbildung 3: eHomeConfigurator: service specification

After firing up 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 4). Until now, no appliances have been specified; there are no electronically controllable appliances in the customer's environment. This means, 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

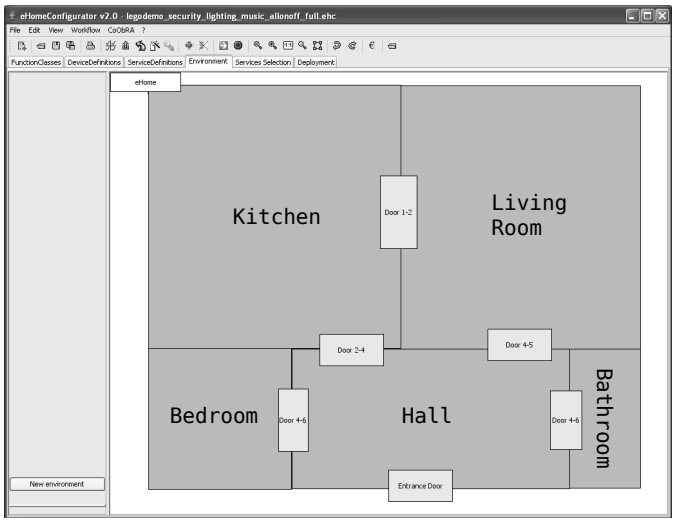

Abbildung 4: eHomeConfigurator: empty environment

(see figure 5). 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`).

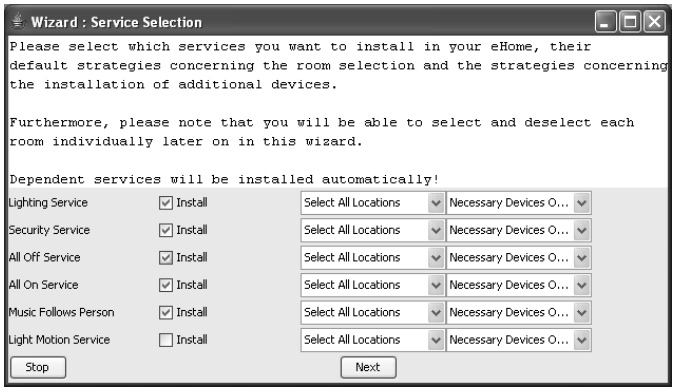

Abbildung 5: eHomeConfigurator: service selection

In the next step, the inhabitant can fine tune the room-selection (see figure 6): Some services like **AllOnService** turns on all the light in the complete environment with one button. are usually only accessed from special locations, like bedroom or hall. They need not to be accessed from every room.

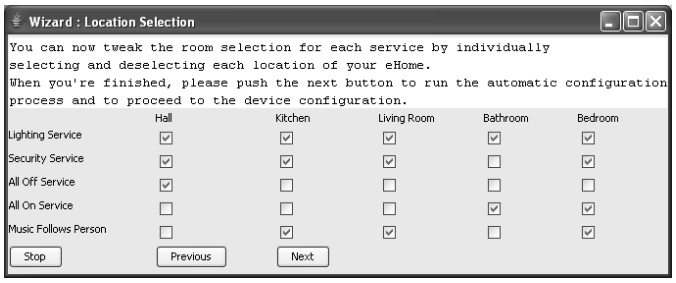

Abbildung 6: eHomeConfigurator: location selection

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, but also alternatives for other drivers, resulting in different required hardware and software components, like hardware used in another demonstrator are possible. In the depicted scenario, also an X10-based realization [X1004] in one or all rooms is possible. The system detects, it can realize the illumination-functionality via an X10-controller service or a Lego Lamp Controller service (see figure 7).

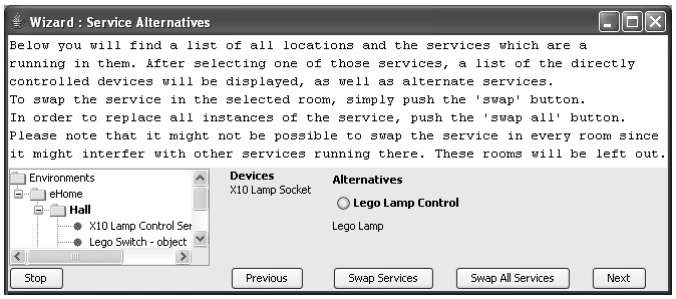

Abbildung 7: eHomeConfigurator: alternatives of software components

When this selection is done, the configuration tool finishes the computation of necessary software components and appliances. The inhabitants can now buy the appliances and enter their addresses and other hardware parameters in our tool (see figure 8). These parameters would be the address numbers of the lego lamps (i.e. 2 for the kitchen), email addresses (i.e. eHome@i3.informatik.rwth-aachen.de) and message texts (i.e. ïntrusion 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 10). On demand, the tool creates a shopping list to indicate which devices have to be bought (figure). 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

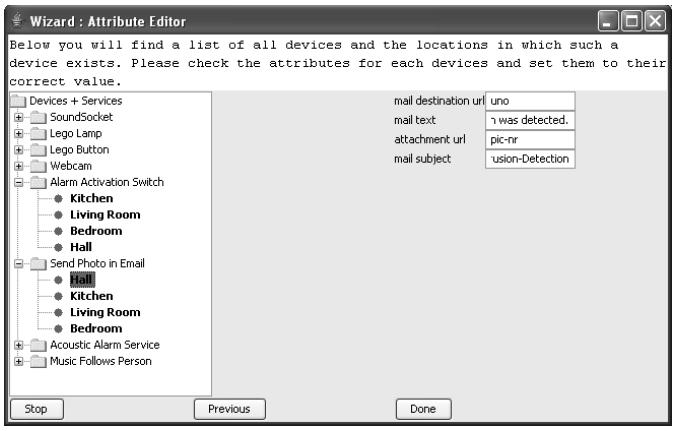

Abbildung 8: eHomeConfigurator: attribute editor

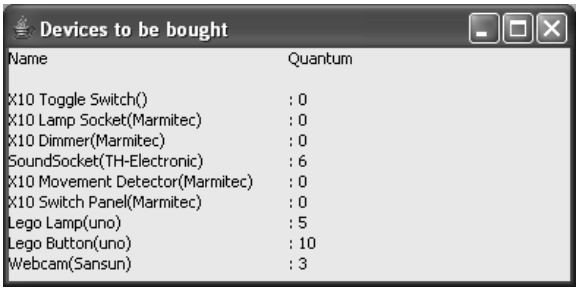

Abbildung 9: eHomeConfigurator: shopping list

deployment tool. This graph is not intended to be processed by the customer, but serves for the automatic deployment and as a visual entry point for debugging. In the context of our scenario, this graph will contain nodes representing the service objects, appliances, and their attributes associated to the different locations. In figure 11, 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 SoundRouter - object), physical contain relations (the Sound Device is in the Living-room), responsibilities of runtime software components for environment elements (Soundrouter - object is responsible for routing sound to the Living-room), and links to attributes (attr edges) and states (state edges).

The whole deployment graph for the lego demonstrator providing the five selected services consists of more than 200 nodes. The initial version of the graph before entering the configuration phase includes only 14 nodes. The XML representation of the complete

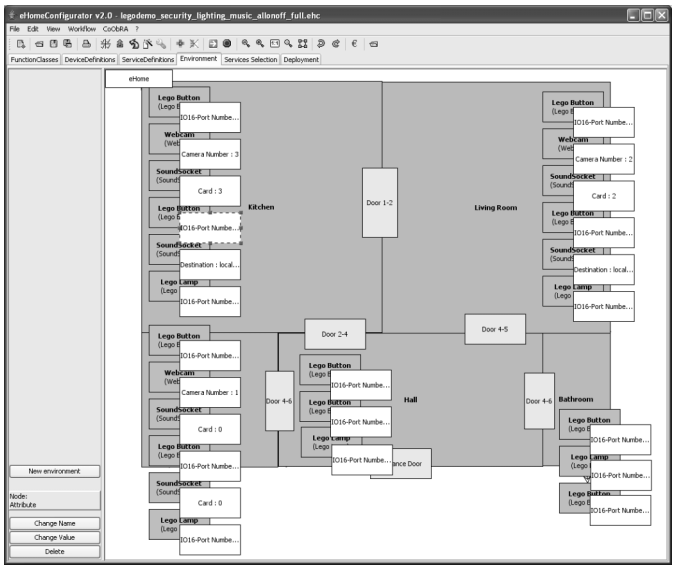

Abbildung 10: eHomeConfigurator: environment with appliances

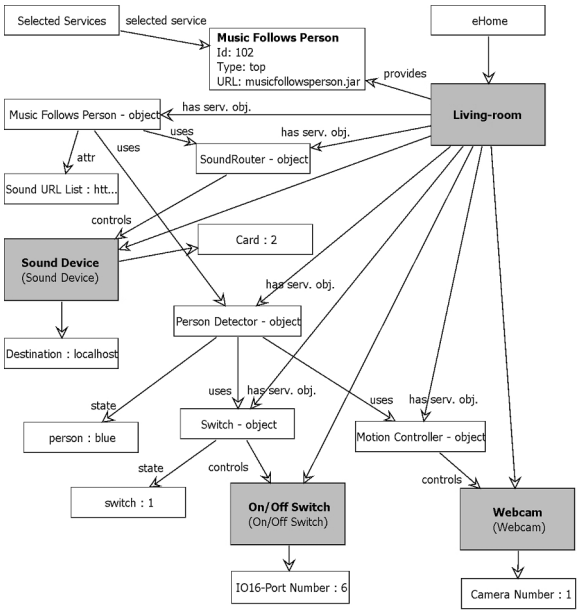

Abbildung 11: Deployable configuration graph for the MusicFollowsPerson Service

configuration graph comprises more than 2200 lines, of which about 800 concern automatically generated deployment information. The rest corresponds to services, devices, and

environment descriptions entered with the help of the specification tool. The configuration phase for the lego demonstrator and the introduced services takes computer science students not familiar with the tool suite less than 15 minutes.

6 Related work

While a lot of research is being done in the field of ubiquitous and pervasive computing, its main focus is usually 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.

6.1 eHome Buildings

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 [inH05]. 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 [TS05] (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.

6.2 Java Context Awareness Framework (JCAF)

The Java Context Awareness Framework, or in short JCAF [Bar05], is a Java framework which offers a set of interfaces to support the developement of context-aware applications. JCAF's main features can be summarized as follows. It is a framework which was designed specifically for event-based applications. This also includes services which usually react to events triggered by sensor-devices in the eHome environment. It supports distributed and cooperating services, i.e. it provides methods of inter-process communication. JCAF also offers a set of security features to restrict access to sensor data and actuator control. It provides methods to verify the origin of incoming events. The last main feature concerns starting, modifying, and stopping software without the need to restart the whole

framework. Even though JCAF and the eHomeConfigurator aim at completely different goals concerning pervasive computing, they can be used in combination.

6.3 SmartHome User Interface

SmartHome User Interface [vDY96] 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.

6.4 ECT and CAMP

ECT [GT04] 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* [THA04] 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.

6.5 Rio

RIO [Den04] is an extension of the Jini-framework [Sun05]. It adds a component model and a component programming model to the Jini-framework. In RIO, components are called Jini Service Beans (JSB). They are described in an XML-based notation called *Operational String*. Operational Strings may include further Operational Strings and Service Bean Elements. Service Bean Elements include all information necessary to install a JSB.

Hence, it is sufficient to specify Operational Strings to deploy a complete component-based system by one operation.

7 Conclusion & Outlook

Automatic configuration of services and appliances is an important requirement for the acceptance of eHome systems. In this paper, we present an approach for simplifying the development process of eHome systems. It addresses the specification, configuration, and deployment phases. We have developed a tool suite, called eHomeConfigurator, to support all these phases of our SCD-process. The eHomeConfigurator calculates and proposes different combinations of appliances and drivers. The resulting configuration graph contains all information needed for an automatic deployment.

We tested our tool not only on the lego demonstrator presented in the introduction but also on other environments including another demonstrator and real buildings [inH05]. Our tool calculated the configurations with negligible time. Even though the search for suitable configurations in this context is proven to be a NP-hard problem [KNS04], the still limited number of nodes poses no problem at this time. However, heuristic algorithms can become necessary in future. The resulting graphs comprised more than 200 nodes for the depicted 5 services in 5 rooms.

In the classical development process, where intense requirements analysis, manual specification and formalization, 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 specification phase specifying functionalities, appliances, and services. The remaining tasks can be carried out by an average customer.

Of course, aspects concerning the user interface can still be improved in future work, localization of appliances and detection (for example with UPnP [UPn00]) of their attributes is ongoing work in the field of pervasive computing. Such extensions should be easily included to our tool suite. Another extension to be tackled is the reconfiguration and multiple middleware support for the deployment tool.

Literatur

[Bar05] Jakob Eyvind Bardram. The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In Hans Gellersen, Roy Want und Albrecht Schmidt, Hrsg., *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, lncs, Munich, Germany, Mai 2005. Springer Verlag. To appear.

[Bei05] Beisheim Holding GmbH. FutureLife – Das Haus der Zukunft. `http://www.futurelife.ch/` (21.08.2006), 2005.

[Den04] Dennis Reedy. Project Rio. `http://rio.jini.org` (10.05.2004), 2004.

[GT04] Humble J. Greenhalgh C., Izadi S., Mathrick J. und 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.

[inH05] inHaus Duisburg. Innovationszentrum Intelligentes Haus Duisburg. `http://www.inhaus-duisburg.de` (21.08.2006), 2005.

[KNS04] Michael Kirchhof, Ulrich Norbisrath und Christof Skrzypczyk. Towards Automatic Deployment in eHome Systems: Description Language and Tool Support. In Robert Meersman und Zahir Tari, Hrsg., *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, Proceedings, Part I*, number 3290 in LNCS, Seiten 460–476. Springer, 2004.

[NSM06] Ulrich Norbisrath, Priit Salumaa und Adam Malik. eHomeConfigurator. `http://sourceforge.net/projects/ehomeconfig`, 2006.

[Ope02] Open Services Gateway Initiative. OSGi Service Platform Specification. `http://www.osgi.org/osgi_technology/download_specs.asp` (2.3.2005), 2002.

[Phi06] Philips. Homelab. `http://www.research.philips.com/technologies/misc/homelab/index.html` (22.8.2006), 2006.

[Sun05] Sun Microsystems, Inc. The Community Resource for Jini Technology. `http://www.jini.org`, 2005.

[THA04] Khai N. Truong, Elaine M. Huang und Gregory D. Abowd. CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In *Ubicomp*, Seiten 143–160, 2004.

[TS05] T-Systems. Wohnen in der digitalen Welt. *Best Practice - Ausgabe für den Mittelstand*, Seiten 17–18, Marz 2005.

[Ulr05a] Ulrich Norbisrath, Priit Salumaa, Adam Malik. eHome Specification, Configuration, and Deployment. `http://ubicomp.org/ubicomp2005/programs/demos.shtml`, 2005. Demonstration Paper D15 on UbiComp2005.

[Ulr05b] 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.

[UPn00] UPnP-Forum. UPnP Device Architecture, Juni 2000.

[vDY96] Rolf Raven Elmar van Dijk und Fredrik Ygge. SmartHome User Interface: Controlling Your Home Through the Internet. *ISES*, 8, 1996.

[X1004] X10. Protocol Specification. `http://www.marmitek.com/en/basisimages/x10_protocol.PDF` (30.5.2005), 2004.