Model-Driven Flow for Assistive Home Automation System Design

Willy Allègre, Thomas Burger, Pascal Berruet

Model-Driven Flow for Assistive Home Automation System Design

W. Allègre * **T. Burger** * **P. Berruet** *

* *Lab-STICC (CNRS), European University of Brittany, Université de Bretagne-Sud, Centre de Recherche, BP 92116, F-56321 Lorient Cedex, France (e-mail: first.last@univ-ubs.fr)*

Abstract: People with disabilities sometimes have considerable difficulties, or even physical incapacities, performing daily tasks independently. Many research works have introduced home automation as a useful way to overcome these activity limitations. However, very few of these accomplishments have focused on the design of intelligent systems which would allow non-experts to model and to adapt a home automation environment for the disabled. This design work is currently restricted to technicians, rather than occupational therapists or others who are able to best understand the needs of those with mobility or cognitive impairments. To take up these challenges, this paper proposes a design flow including a component approach for modeling system architecture and a range of services to meet the needs of both developers and users. Based on model driven engineering, it automates the control code generation for home automation systems.

Keywords: home automation systems, discrete event systems, engineering methods for human machine systems, model driven engineering, assistive technologies, human-centered design

1. INTRODUCTION

People with disabilities sometimes have considerable difficulties, or even physical incapacities, performing daily tasks independently. Whether they are disabled physically, intellectually, or due to sensory impairment, their limited capabilities do not allow them to control the large range of home devices. In the new disability definition of the World Health Organization, contextual factors are introduced in order to consider the impact of the environment on the person's functioning (Schneidert et al. (2003)). Home automation is one possible solution to overcome these disabilities, enabling one to perform daily tasks without assistance from caregivers or relatives. Therefore, Home Automation Systems (HAS) must be able to automate and provide users with adapted and easy-to-access services, including high-level interaction as if a home were smart.

Smart homes join the field of ubiquitous computing (Weiser (1993)) in which information processing and communication has been thoroughly integrated into everyday objects able to offer its own services. Due to the massive integration of smart devices in the home, it becomes more and more complicated for users to ensure the control of these objects. Various studies have focused on the flexible composition of high-level services in order to make life easier. Software architecture is often based on a multi-agent system considering both user needs and context information to dynamically adapt the process of service composition (Kushwaha et al. (2004)). Other studies deal with goal-based interaction (Herfet et al. (2001)), where user requirements are translated by an objective that the system has to reach. Furthermore, works from knowledge engineering focus on the formal representation of home automation systems using ontologies. Firstly defined as "an explicit specification of a conceptualisation" (Gruber et al. (1995)), an ontology is basically a way to store, organize, and represent knowledge. In (Reinisch et al. (2008)), ontologies provide an abstract view of heterogeneous network infrastructures in order to accommodate all functionality found in building automation systems. Making explicit the semantic relations and the nature of domain concepts, ontologies can be used to automatically generate new knowledge. A context management system (Kim and Choi (2008)) can, for instance, take advantage of such logical relationships to deduce context information from current information using a rule-based inference reasoner.

These initiatives are recent and show the dynamics of home automation dedicated to persons with disabilities. However, very few of these accomplishments (Noury et al. (2003)) have focused on the design of intelligent systems. In the context of Ambient Assisted Living, needs in terms of adaptation are constantly evolving. For financial reasons, the systematic intervention of an expert can not be considered. Moreover, it seems natural that the adaptation of a living space for a wheelchair-bound person should be made by occupational therapists (OT) or any other person intimately related to the user (e.g. a familly member), rather than by technicians.

This paper presents a modeling approach to assist designers in building an assistive HAS. In section 2, we introduce the two roles (designer and user) included in our human-centered design process in order to clearly define the challenges that have to be overcome. Then, section 3 proposes a modeling method adapted to the designer's point of view, based on model-driven engineering and adopts a component-based approach for the design of a home automation environment. To consider user requirements

as well, in section 4, we introduce an interaction model, based on services and modes. Finally, section 5 gathers all of the previous parts into an illustrated model-driven flow so as to informally validate the adopted design approach.

2. A HUMAN-CENTERED DESIGN PROCESS

Two main roles can be identified in our work, that of the designer and that of the user. The **designer** is not an expert, but a person sensitive to home automation, primarily responsible for building an environment which is suited to **users** with disabilities. Thus, the designer is, above all, someone who is able to properly consider the context of living with a disability (e.g. an OT).

From the user's point of view, high-level interactions are essential for two reasons. First, the effective management of several devices at the same time becomes increasingly difficult for the user. By providing intuitive interactions, this cognitive load [1] can be reduced: "Leave" presents a higher semantic level than "Open the door". However, these interactions can above all compensate for physical disabilities by facilitating or automating service activation (Truong et al. (2009)): the "Leave" composite service provides the activation of several basic services such as "Open the door", "Switch off lights" with a single control.

In addition, HAS must be easy to design, easy to maintain, easy to control, and flexible. All these requirements make the use of computer-aided design tools essential for the designer. Paradoxically, there are few tools and methods for high-level design adapted to these complex systems. They are currently developed by using low-level procedures and methodologies that do not allow for the specification of system functionality without undertaking its implementation on a specific technology platform. Proprietary solutions (Echelon Corporation (1998); KNX Association (1993)) are technology dependent and the syntax they provide is often limited and not very intuitive. In this case, designers spend more time dealing with syntax, than actually being able to look thoroughly into the functional part of the design. Thus, the installation of such systems is left to home automation experts.

The objective of this work is twofold. First, we would like to propose a design method for controlling HAS which has an abstraction level high enough to become accessible to a non-expert. Secondly, the dependable designed system must provide high-level interactions to satisfy the needs of persons with disabilities.

Overall, our goal is to provide a sufficient level of abstraction so as to include both roles in the HCD process (Human-Centered Design, ISO 13407 compliant, Maguire (2001)) shown in figure 1. Once the user's requirements are collected, the designer (ideally OT) can specify the context of use through a HAS environment model composed of many devices. The OT can, thus, take advantage of our interaction model wich is rich enough to adapt the environment to the user's defined needs.

[1] The cognitive load can become excessive due to the simultaneous management of several devices and the lack of automation (intrinsic load) or the way information is presented (extraneous load) (Paas et al. (2004)).

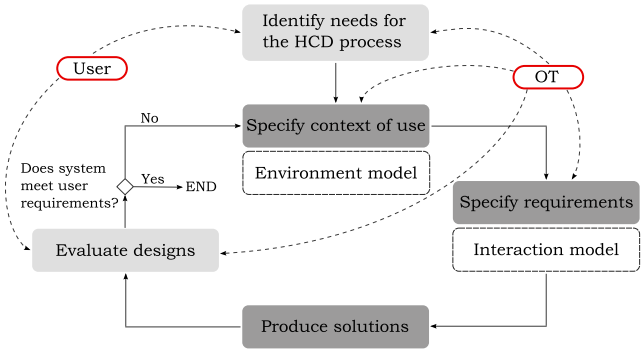

Fig. 1. Human-centered design process for assistive HAS

3. ENVIRONMENT MODELING

In this section, we consider the designer's point of view, who is in charge of building a home automation environment. Firstly, we propose applying model driven engineering concepts to assistive HAS in order to make the design accessible to non-experts. Secondly, bottom-up modeling by component aggregation is proposed to make this design intuitive. Thus, we introduce points of view and concepts defined in the proposed model.

3.1 Model Driven Engineering

Model driven engineering stands out from the different software engineering approaches. Its approach assumes that all or part of a computer application is generated from models. MDA (Model Driven Architecture) (Miller et al. (2001)) provides tools, concepts, and languages to create and transform models, developing a large community around this area.

This original approach is based on three fundamental concepts: model, metamodel, and model transformation. A model is a partial representation of a physical system which can be defined through a specific description language named Domain Specific Language (DSL). The metamodel defines the language structure (grammar) used to build models (any model conforms to a metamodel). Finally, the model transformation defines rules to translate a source model into one or more target models.

A domain specific model allows the designer to focus on domain specification rather than system implementation. By increasing the level of abstraction, the design becomes intuitive and, therefore, more accessible to non-experts in these technologies. Following the MDA paradigm, the implementation is then automated through the use of model transformations. From a Platform Independent Model (PIM) using an appropriate language (e.g. a DSL), it can generate one or more Platform Specific Models (PSM) to generate the functional code of the system.

We believe that the joint use of a DSL and a model driven architecture can provide a level of abstraction sufficient to make assistive HAS design available to a non-expert person. First, the automatic generation of the control code of the different devices is a key point in the home automation design process. It must cope with the diversity of protocols and media (KNX/EIB, Bluetooth, infrared, etc.) associated with the range of home devices. Model transformations provide a solution: from domain PIM, it

can generate PSM from which the control code specific to a technology is generated. We also propose simplifying the design of a HAS with a semantic high-level modeling not only to facilitate the designer's work, but also to limit his need for technical skills. The ability to manipulate a model dedicated to home automation allows the designer to sidestep any constraints related to the handling of a specific programming language.

He can focus both on the living space modeling and the needs of potential users. Such modeling is made possible by the use of a domain-oriented description language. Whether for generating/adapting user interfaces (Mukasa et al. (2005)), or dedicated to the control part of home automation systems (Jimenez et al. (2009)), the use of such a language facilitates the selection and specification stages through a visual and intuitive system representation.

3.2 Component-based approach

The component approach introduced in the CBSE (Component Based Software Engineering) paradigm defines a component as a "black box" which displays only its interfaces and requirements (Szyperski et al. (2002)). A component is considered as substituable, reusable, and composable (components can be combined to obtain more complex functions).

Previous works from FMS (Flexible Manufacturing Systems) focus on bottom-up modeling by component aggregation for conveying systems (Lallican et al. (2007)). In this work, a software component, directly related to a physical component, is composed of operations and "views" (*control*, *operating part*, *constraints*, etc.). The real purpose of the component is its use as a building block to design a conveying system and its control part.

Here, we propose applying such an approach to the field of home automation. This *bottom-up* approach is adopted because the definition of a component from level N should be done knowing operations of components from level $N-1$, in order to couple it with highest level services. The aggregation strategy falls under the responsability of the designer, who has to take into account the possible reuse of aggregate components. Indeed, once configured, they can be reintegrated into the library with a reuse perspective.

The environment model we propose distinguishes between resource and system points of view. Their definitions are naturally inspired by FMS concepts and take up their terminology. From the **resource** point of view, an **operation** allows one to change the states of a resource. The latter is a software representation of a home device that has one or more operations. For instance, the resource *TV1* can perform several operations, such as *TurnON TV1*, *Mute TV1*, etc. For reasons explained below, the states of each resource can also be submitted to **restrictions**. Resources are logically implemented by components based on a standard structure. The **system** point of view, defined to raise the level of abstraction, is responsible for supervision of the HAS. From this point of view, a **function** is defined as an abstract concept, which when implemented by a resource, provides an operation. Moreover, a function, can be carried out by various resources. For instance, *Lighting* function can be implemented by the resource *Lamp1* to achieve the *Lighting the Lamp1* operation. The restrictions defined at the resource level are then transmitted by **inheritance** to the system level.

Table 1. Points of view and related concepts

Resource	**System**	**User**
Operation	Function	Service: "I want"
Restriction	Inheritance	Mode: "I forbid"

Unlike FMS, assistive HAS is above all dedicated to the person. That is why we introduce a third point of view, the one of the **user**, to separate the physical model based on components from the logical model considering user specific needs. Operation and function are expressed in terms of **service** whereas restriction is expressed in terms of **mode**. All these concepts are summarized in table 1 and the user point of view is detailed in the next section.

4. INTERACTION MODELING

In this section, we consider the way a user should be able to interact with his environment, to allow the designer to model interactions between the user and the environment regardless of the human-machine interface that will be used. The proposed model allows user to interact with the home automation environment in two different ways. One allows for the expression of the injunction "I want" through services. To complement this, the second one allows for the expression "I forbid" implemented by modes. First, we consider the expression of prohibitions, then we detail the request for services.

4.1 Modes

We call **modes** a restriction on the state space of a resource. Each mode (user point of view) is interpreted by an evolution space E_r (resource point of view). Through modes, the designer can restrict the overall state space $E_{r_0} = \{e_1, e_2, e_3, e_4\}$ of a resource defining a prohibited state, for example e_3. In this case, the states $e_i \in E_r$ potentially accessible from a resource r are defined in (1), where $\overline{E_r}$ represents the states of E_{r_0} which is forbidden and inaccessible by switching the mode.

$$\begin{aligned} &\overline{E_r} = e_3 \\ &E_r = E_{r_0} \setminus \overline{E_r} = \{e_1, e_2, e_3, e_4\} \setminus \{e_3\} = \{e_1, e_2, e_4\} \end{aligned} \quad (1)$$

In addition, several restrictions on the state space can be activated simultaneously with as many modes. This allows for a more precise restriction on the evolution space of a resource. For example, if E_r^1 represents the state space allowed by a first mode (e.g. *energy conservation*) and E_r^2 represents the state space allowed by a second mode (e.g. *child unattended*), then the state space allowed by these two modes is calculated by the intersection of E_r^1 and E_r^2, as illustrated in (2).

$$E_r = E_r^1 \cap E_r^2 = \{e_1, e_2, e_3\} \cap \{e_2, e_3, e_4\} = \{e_2, e_3\} \quad (2)$$

Mode handling, well-tried in the field of FMS, has been adapted to be applied to the assistance of people with disabilities. In (Hamani et al. (2006)), modes are defined by a set of states characterizing a resource according to a point of view. States that are grouped under one mode are mutually exclusive (e.g. among the *normal*, *degraded* or *out-of-order* states of the *functioning mode*, a single one

can be active). Modes are no longer defined in relation to production, but are made flexible to meet the needs of the user, thus they allow designers to model prohibitions on a home automation environment.

However, the definition of modes is not only limited to constraints on resources. A manager guarantees coherence between the modes, ensuring not only that the state of a resource is no longer attainable, but also that it leaves this state, if by any chance it was the current state when changing modes. For instance, in *evacuation* mode, the designer defines that the doors must be opened. In other words, it prohibits the *closed* state of the doors: *Syst-Forbids(close, door)*. Accordingly, the operation associated with the closing is no longer available. If in our example, a door is closed when changing modes, the system must force it to open.

4.2 Services

Beyond the expression of prohibition, it is necessary, in order to consider the diversity of needs that people with disabilities can express (De Lamotte et al. (2008)), to also propose a model for the expression of queries on the resources. This is what we call the **service** model.

Therefore, we first define a service as a task offered by the system to the user in order to meet his needs. Unlike operations and functions, services are defined from the user point of view (see table 1). It then seems relevant to distinguish how the service is activated from its semantics. This distinction can lead to a definition where services are defined by the pair (*activation*, *semantic level*). A service S can be activated in different ways ($c1$, $c2$, $c3$):

- $c1$: under the request of the user at runtime
- $c2$: automatically under a predetermined condition
- $c3$: automatically with user confirmation

A mode switch may involve, by activation of type ($c2$), triggering a service to put resources in accordance with this mode. A Service S can also be differentiated according to the semantic level it conveys ($n1$, $n2$, $n3$):

- $n1$: basic service related to the operation of a resource
- $n2$: scenario composed of several level $n1$ services
- $n3$: semantic high-level service related to a function

All or part of these definitions are illustrated through the case study in the following section.

5. DESIGN FLOW: A CASE STUDY

In this section, we summarize all the previous parts through the general methodology adopted for an assistive HAS design. A key stage before going into system modeling is collecting the user's specific needs. This part of the human-centered design process (figure 1) is the responsability of occupational therapists and medical staff. The second stage is the system modeling itself as described above, using an environment model to specify context of use, an interaction model to specify requirements, and automatic control code generation to produce design solutions. Each step of the model-driven flow shown in figure 2 is discussed and illustrated to informally validate the design approach adopted in our work.

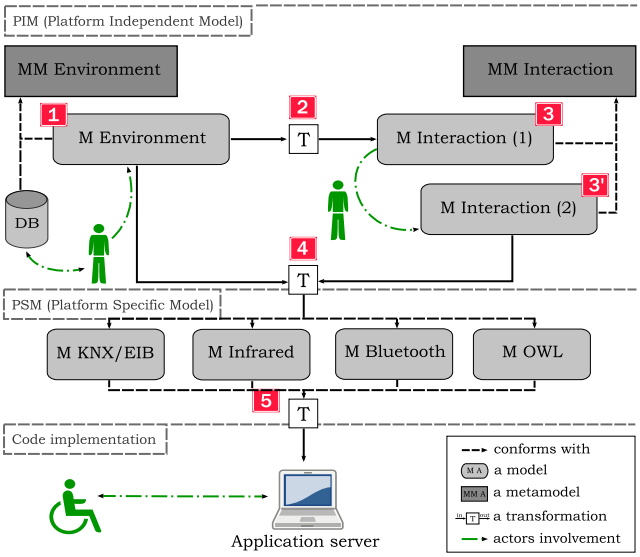

Fig. 2. Proposed model-driven flow for an assistive HAS

Case Study: The user in this case study is a person, who has been hemiplegic for a few months, after a stroke: the first leading cause of disability among adults in the United States and Europe [2]. The latter has limited capabilities (walking difficulty and a non-functional upper limb) that do not allow him to perform daily tasks without assistance of caregivers or relatives. Due to this mobility impairment, he only walks for transfers from wheelchair to bed, and vice versa. After being hospitalized in a stroke unit, he is now in a rehabilitation center within the framework of a full-time hospitalization. Finally, this patient also has cognitive impairments emerging in the context of Alzheimer type dementia, diagnosed before the stroke [3]. As these cognitive impairments decrease his autonomy, the home automation environment must be able to adapt to his emerging needs.

5.1 Environment modeling by components aggregation

The designer, non-expert in the home automation field, builds his environment by manipulating an independent domain specific model. The originality lies in a bottom-up modeling approach by successively aggregating components until one represents the whole system.

Case Study: Healthcare center rooms are equipped with KNX technology and have many devices controllable via infrared remotes. Figure 3 shows the plan of the user's apartment that the occupational therapist seeks to model. Starting with the basic components from a database (*C1*, *TV1*, *L1*, etc.), he decided to define a highest level component on a room scale (*Living room*), associating itself with many other components to form the component *Apartment*. These aggregate components, once defined and configured, can be stored in a library for later reuse.

[2] Disability (physical or cognitive) affects 75% of stroke survivors (Diener and Wong (2009)).

[3] Alzheimer disease (AD) is the most common form of dementia and affects about 26 million people worldwide (Brookmeyer et al. (2007)). Moreover, recent studies show a link between AD and stroke (Nagai et al. (2009)).

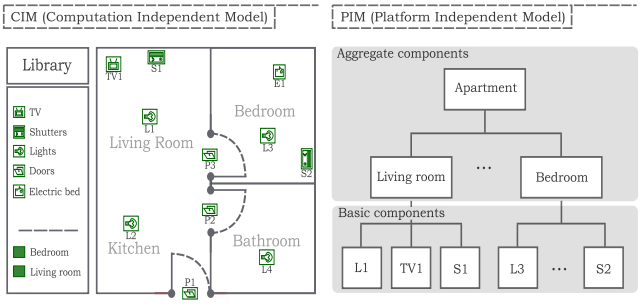

Fig. 3. Home automation environment modeling

5.2 Towards user-specific interaction model

As explained in section 4.2, services offered to the user, regardless of their semantic levels and their activation process are described using operations. Similarly, modes are described in terms of states, reachable or not. With this description indicating the correspondence between operations-services and states-modes, it is possible to move from an environment model describing the system architecture to an interaction model specific to the user.

```
1   module Env2Int;
2   create OUT : MMInteraction from IN : MMEnvironment;
3
4   rule operation2service {
5       from
6           s : MMEnvironment!Operation
7       to
8           t : MMInteraction!C1_Manual_Activation,
9           u : MMInteraction!N1_Basic_Service (
10              Name ← s.name,
11              Service_has_activation ← a
12          )
13  }
```

Fig. 4. Env2Int transformation excerpt, written in ATL

Case Study: Operations *SwitchON* and *SwitchOFF*, described above, are transformed into basic services *UserWant(SwitchON, L1)* and *UserWant(SwitchOFF, L1)*. Figure 4 gives an excerpt of the *Env2Int* model transformation using ATL (Atlas Transformation Language) (Jouault et al. (2006)). It transforms an environment model into an interaction model (line 2). A single rule is shown: rule *operation2service* (lines 4-13), which transforms a resource operation into a basic manually activated service, i.e. a $S(c1, n1)$ service (section 4.2).

5.3 Interaction modeling according to the needs of the user

From the basic services of the previous step, it is possible to create composite services (scenarios) which can be submitted to any conditions for activation. The same goes for mode definitions.

Case Study: In a first step, the user mainly needs automation services with useful scenarios to compensate for his physical impairment. The OT defines, for instance, a "Leaving home" scenario which can be activated manually ($S(c1,n2)$) including *switching off multimedia devices*, *opening door*, *switching off lights*, *closing door*, interesting mainly for effort and time-saving reasons (De Lamotte et al. (2008)). Moreover, the user's needs change a few months later, with the predictive increase of cognitive impairments (disorientation, fugues, etc.). In addition to automation services, the OT defines prohibitions on the home automation environment in order to reduce the risk of runing away and ensure patient safety. "Anti-runaway" mode forbids users from opening the door during the night: *SystForbids(open, D1)*. On the other hand, the resolution of a semantic high-level service is a more complex problem requiring the use of constrained optimization programs, or methods from artificial intelligence. Unlike services with $n1$ or $n2$ semantic level, high-level services ($n3$ semantic level), such as *increasing light* or *decreasing noise* is asked without specifying the resource involved. These aspects are beyond the scope of this paper. However, we believe that a representation of knowledge such as an ontology, may be interesting for describing and reasoning on context information (Kim and Choi (2008)) and user activities. We give some leads in the conclusion.

5.4 From independent DSL to Platform Specific Models

From environment (step 1) and interaction (step 2) PIM, it is now possible to generate a PSM. Danah (Lankri et al. (2008)) is the chosen target platform, which has been metamodeling to be integrated in our design flow. It is a middleware dedicated to ambient assisted living, allowing both environmental control and wheelchair navigation. Each component from the environment model has a standard structure and consists of its own operations. Therefore, it is possible to generate PSM elements and associated Danah *units* from hardware specification of this model. Thus, interaction model allows to transform previously defined interactions into Danah *actions* in a manner consistent with the platform specific metamodel.

Case Study: For example, *L1* is an instantiation of the *Lamp* component, it has operations *SwitchON* and *SwitchOFF*. Hardware specification properties give information about the technology involved to control the latter: *KNX/EIB*. From this information, one *unit* L1 with the specified protocol *EIB*, the associated library *libeibd*, and the associated command is automatically generated.

5.5 Control code generation for the execution platform

From the PSM described above, pseudo-code conform to Danah syntax is then automatically generated. This process is made using a model to text transformation. The resulting control code is physically launched in several servers spreading previously defined interactions. The hardware architecture used in our work is a decentralized one where servers communicate among themselves to deliver services. Users can connect to servers via a PDA terminal, most often embedded in their wheelchairs, to solicit services.

6. CONCLUSION & PROSPECTS

An original design flow for assistive home automation systems has been presented in this paper. According to the human-centered design process, it adopts a mixed modeling approach to meet the needs of both the designer and the user. First, the high-level modeling, which uses

a domain specific language and automatic control code generation, makes the design of such systems available to non-experts. A hierarchical environment modeling based on component aggregation is coupled with interaction-oriented modeling to meet the needs of persons with disabilities. Thus, modes along with services with different semantic levels, offer a comprehensive model to enhance the autonomy of these persons. Based on MDA, this design flow offers a flexible and evolutive solution for the integration of new home automation technologies. In a home automation environment, specifically dedicated to persons with disabilities, knowing who is where and what they are doing is central to enabling adapted intelligent behavior. We think ontologies can be useful to model context information especially user interaction. The resolution of semantic high-level services can benefit from such a knowledge representation. For this, the model-driven flow can adopt the ODM (Ontology Definition Metamodel) specification to make the concepts of Model-Driven Architecture applicable to the engineering of ontologies (Consulting and Deere (2005)). Finally, in the longer term, a full-scale deployment is planned at Kerpape MFRRC [4] to assess the work presented in this paper.

ACKNOWLEDGEMENTS

This work is financed by the Brittany Region and is the result of a cooperation supported by Vity Technology (www.vity.com) and Kerpape MFRRC.

REFERENCES

Brookmeyer, R., Johnson, E., Ziegler-Graham, K., and Arrighi, H. (2007). Forecasting the global burden of Alzheimers disease. *Alzheimers Dement*, 3(3), 186–191.

Consulting, D. and Deere, J. (2005). Ontology Definition Metamodel.

De Lamotte, F., Departe, J., Le Saout, F., Diguet, J., and Philippe, J. (2008). Quatra: Final report (technical report, in french). Lab-STICC. See demo here: http://www.youtube.com/watch?v=T6GCFnkLTc0.

Diener, H. and Wong, P. (2009). Developments in Secondary Stroke Prevention. European Neurological Review - Volume 3 Issue 2.

Echelon Corporation (1998). Lonmaker Integration Tool. Official website: http://www.echelon.com/.

Gruber, T. et al. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5), 907–928.

Hamani, N., Dangoumau, N., and Craye, E. (2006). A functional modeling approach for mode handling of flexible manufacturing systems. In *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing*, 271–276.

Herfet, T., Kirste, T., and Schnaider, M. (2001). EMBASSI multimodal assistance for infotainment and service infrastructures. *Computers & Graphics*, 25(4), 581–592.

Jimenez, M., Rosique, F., Sanchez, P., Alvarez, B., and Iborra, A. (2009). Habitation: a domain-specific language for home automation. *IEEE Software*, 26(4), 3038.

[4] Kerpape MFRRC: Kerpape Mutualistic Functional Reeducation and Rehabilitation Center (http://www.kerpape.mutualite56.fr/)

Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., and Valduriez, P. (2006). ATL: a QVT-like transformation language. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 720. ACM.

Kim, E. and Choi, J. (2008). A Context Management System for Supporting Context-Aware Applications. In *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008. EUC'08*, volume 2.

KNX Association (1993). Engineering Tool Software. Official website: http://www.knx.org/.

Kushwaha, N., Kim, M., Kim, D., and Cho, W. (2004). An intelligent agent for ubiquitous computing environments: smart home UT-AGENT.

Lallican, J.L., Berruet, P., Rossi, A., and Philippe, J.L. (2007). A component-based approach for conveying systems control design. In *Proceeding of the 4th International Conference on Informatics in Control, Automation and Robotics ICINCO*, 329–336.

Lankri, S., Berruet, P., Rossi, A., and Philippe, J. (2008). Architecture and models of the DANAH assistive system. In *Proceedings of the 3rd international workshop on Services integration in pervasive environments*, 19–24. ACM.

Maguire, M. (2001). Methods to support human-centred design. *International journal of human-computer studies*, 55(4), 587–634.

Miller, J., Mukerji, J., et al. (2001). Model driven architecture (mda). *Object Management Group, Draft Specification ormsc/2001-07-01*.

Mukasa, K., Ziegeler, D., and Zuehlke, D. (2005). A Model-based approach for useware development. In *Proceedings of the 16th IFAC World Congress*.

Nagai, M., Hoshide, S., and Kario, K. (2009). Hypertension and dementia. *American journal of hypertension*, 23(2), 116–124.

Noury, N., Virone, G., Ye, J., Rialle, V., and Demongeot, J. (2003). New trends in health smart homes. *ITBM-RBM (RBM)*, 24(3), 122–135.

Paas, F., Renkl, A., and Sweller, J. (2004). Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional Science*, 32(1), 1–8.

Reinisch, C., Granzer, W., Praus, F., and Kastner, W. (2008). Integration of heterogeneous building automation systems using ontologies. In *34th Annual Conference of IEEE Industrial Electronics, 2008. IECON 2008*, 2736–2741.

Schneidert, M., Hurst, R., Miller, J., and Üstün, B. (2003). The role of environment in the International Classification of Functioning, Disability and Health (ICF). *Disability & Rehabilitation*, 25(11), 588–595.

Szyperski, C., Gruntz, D., and Murer, S. (2002). Component software: beyond object-oriented programming. 2002.

Truong, T., de Lamotte, F., Diguet, J., et al. (2009). Proactive remote healthcare based on multimedia and home automation services. In *Proceedings of the 5th IEEE CASE*, 385–390.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36, 75–84.