A design process enabling adaptation and customization of services for the elderly

Jean-Baptiste Lezoray, Maria-Teresa Segarra, An Phung Khac, André Thépaut, Jean-Marie Gilliot, Antoine Beugnard

A Design Process Enabling Adaptation and Customization of Services for the Elderly

Jean-Baptiste Lézoray, Maria-Teresa Segarra, An Phung-Khac
André Thépaut, Jean-Marie Gilliot, Antoine Beugnard

Institut Télécom, Télécom Bretagne, Département Informatique,

Technopôle Brest-Iroise - CS83818 - 29238 Brest cedex 3, France

{jb.lezoray, mt.segarra, an.phungkhac, andre.thepaut, jm.gilliot, antoine.beugnard}@telecom-bretagne.eu

Abstract

In the next decades, the growth in population ageing will cause important problems to most industrialized countries. To tackle that issue, Ambient Assistive Living (AAL) systems can reinforce the well-being of elderly people, by providing emergency services, autonomy enhancement services, and comfort services. Those systems will postpone the need of a medicalized environment, and will allow the elderly to stay longer at home.

However, most elderly have specific needs and deployments of such systems are likely unique. Furthermore, the needs evolve in time, and so does the environment of the system. Making this observation, we propose the use of a design method, the medium approach, to enable the dynamic adaptation of AAL systems. We also study the possible customizations of the design method to make it more suited to the AAL domain.

1 Introduction

In the next decades, most industrialized countries will face important problems with the growth in population ageing. Statistics data provide a clear picture of the problem dimension. According to results of the SHARE survey [2] (Survey of Health, Ageing and Retirement in Europe) funded by the European Commission, in 2050 the share of the above sixty-five age group will be around 28%, instead of 16% nowadays. According to [20], the number of octogenarians in France was tripled since 1960, and will be again multiplied by 2.4 by the 2050.

Specialized institutions will not be able to handle that situation without any alternative solution. One of them is the use of Ambient Assistive Living (AAL) services that aim at enabling people to stay at home for a longer time, delaying the need for a specialized, medicalized institution. Those services are therefore mainly health-oriented services. However, according to [25], elderly living alone and having bad health feel ten times more lonely than others. The study also shows a direct link between a bad health and the fact that they feel lonely. Similarly, Gaymu *et al.* in [28] identify the main factors for an elderly to move to a nursing home : *a)* becoming disabled, *b)* having no living spouse or child, and *c)* having no close family. That leads us to the conclusion that having AAL services dedicated to create or maintain social links with the relatives is almost as important as health-oriented services.

Most AAL applications are either too general or too specific. A too general AAL application provides services not well suited to final users. Each elderly has specific diseases, capabilities and habits, and the system should consider them independently from the provided services. Similarly, the needs of the elderly may evolve over time (e.g., between night and day, or between winter and summer), and the application environment is different between each deployment. On the other hand, if an AAL application is too specific, only a very few people will use it due to its inadequacy. Moreover, its implementation will restrict its use to very specific target

platforms and deployments. In the context of a deployment at a large scale, designing customizable and adaptive AAL systems is a solution to tackle those problems.

We are currently working on a distributed AAL system for the elderly that offers a set of services (health oriented, communication-oriented, and information oriented) allowing them to live longer at their preferred place in a safer way, and to maintain social links with their relatives and their circle of friends. Our research aims at defining a design methodology that eases the development of configurable and auto-adaptive distributed applications. In our methodology, we identify a set of concerns that should be taken into account when developing such applications, including the necessary mechanisms to ensure architectural reconfiguration capabilities that make an application able to dynamically switch between configurations. We will extend it to make it suitable for the specific constraints of AAL systems.

This work is part of the SIGAAL project [26, 27], which aims at reducing loneliness among elderly people, by providing services that allow, for low cost, maintaining or even strengthen social links and detecting emerging vulnerabilities of the elderly.

The paper is organized as follows. The next section analyses the specific needs of adaptation in AAL systems for elderly people through some examples, and introduces the formal basic fonctionalities that should be implemented to build an adaptive software. The third section presents the background used for our solution, which is the generic concept of *adaptive medium*, and its underlying design process which is based on Model Driven Engineering techniques. The fourth section studies how to extend that method to make it applicable to the specific needs of AAL systems, especially by introducing heterogeneity concepts. Finally, sections five and six positions our work in the research results concerning AAL and adaptation, and discusses future work, respectively.

2 Adaptation of AAL systems for the elderly

Ambient Assisted Living systems denotes the use of computing technologies to improve the being of users. The services provided by indoor AAL systems can be classified in three subdomains [18] :

- Emergency treatment services aims at predicting, detecting and/or preventing unsafe situations by propagating alerts in emergency situations (e.g., sudden falls, heart attacks).
- Autonomy enhancement services aim at providing tools to postpone the need of assistance by health caregivers or relatives (e.g., a cooking assistance system).
- Comfort services are services that greatly enhance the quality of life (e.g., logistic services, infotainment services, or services to maintain or create social links).

We claim that AAL systems for the elderly must be adapted to fit each situation as each elderly has specific needs, and each deployment has also strong specificities. Each elderly has specific :

- **Diseases**: an application will neither provide the same services nor behave the same if the system is deployed for a visually impaired person or for a person with muscular problems: the visually impaired person must have an interface with specific fonts, colors contrast, or even a vocalization of the provided services, and the second one must have services with specific simplified control interfaces. The capabilities of each individual must also be taken into account.
- **Acceptability**: to be efficient, the system has to be perceived positively by the elderly people and by the circle of persons in charge of them (relatives, medical staff...). The acceptability of such a system has to be taken into account. For some people, having a camera analyzing their moves and position is inappropriate due to privacy issues, even for a very useful service. For others that option is not problematic. Therefore, the AAL system must be customizable depending of the acceptability.
- **Habits**: the application should dynamically adapt to the habits of the elderly by providing tailored services.
- **Needs depending on the time**: some services must be adapted to the evolution of the elderly

needs over time, e.g., between night and day, winter and summer, or in the case of a progressive disease such as Alzheimer.

Moreover, each deployment has also specificities:

- **Target platforms**: to avoid being bounded by a specific platform (e.g., a specific set-top box[1]), the application should be runnable on different target platforms.
- **Home configuration**: number of rooms, network configuration, layout of the devices...
- **Available resources**: home automation captors and actuators, available devices (TV, tablet PC...), each one having different capabilities in terms of memory, computing capabilities...
- **A dynamic environment**: the environment is not static, and available resources may evolve over time, e.g., by an upgrade or due to a dysfunction.

A combination of each of these specificities leads to a likely complete customization of the AAL system and of its provided services for each deployment. Considering those constraints, we strongly believe that such AAL applications must be distributed, customizable, adaptable, and heterogeneous.

The **distributed nature** of AAL applications is due to *a)* the multiplicity of the location of the access points (for medical staff, for the elderly, for the relatives), *b)* the pervasive nature of such systems, and *c)* security reasons : having a single central point is not a reasonable solution as there is a strong need of quality of service, a distribution and/or replication of the internal parts of the application is a common (partial) solution to settle this.

The **customizable nature** of AAL applications is due to the fact that *a)* applications should be able to run on devices with limited capacity to reduce the costs; in order to reduce the need of resources, they must be tailored to each particular device, and *b)* each target environment is likely unique so a single solution would be too specific.

[1] A set-top box is a programmable hardware connected to a TV set and managing the display on it.

The **adaptive nature** of AAL systems is related to the dynamicity of the environment and the elderly needs and habits. The application should be able to dynamically adapt whether for a short time scale (e.g., adaptation of the font scale for an elderly reading news on TV, or an emergency mode) or for a long time scale (e.g., adaptation of some services between winter and summer, such as a temperature monitoring system). Moreover, there is also an inherent need of adaptation due to the distributed nature of such an AAL system, and to the need of balancing between the non-functional requirements (resource preservation, device availability, network charge balancing...).

The **heterogeneous nature** of AAL applications is related to their distributed nature. As the internal parts of the applications may run on different target platforms (e.g., component-based, aspect-based, script-based...), the capabilities of each of them must be taken into account, including the heterogeneity of the adaptation mechanisms and capabilities.

3 Homogeneous Adaptive Applications

The specification of a dynamically adaptive application must tackle two major difficulties. The first one is the **specification of consistent variants** of an application, considering the common parts and the variations. As a distributed application is composed of multiple internal parts that collaborate to provide a service, the design and distribution of the internals of variants of such an application is a hard task. The second one is the **specification of the transitions** between the variants of the application, leading to an adaptation plan deduced either by computation or by specification of transition assets. This is also a hard task as each transition comprises multiple changing tasks that could be either architectural, configurational, and/or parametrical. Moreover, the data and the state of the application have to be transferred to the new variant. During an adaptation in a distributed application, those actions will also have to be distributed and coordinated.

To build such an adaptive application, Phung-Khac *et al.* propose a methodology that allows the adaptation of a distributed application [21]. It is based on a development process by Kaboré *et al.* which allows the specification of multiple variants

of a distributed software [16]. It provides a framework where the software is produced by applying a set of successive refinements based on a MDA design approach [19]. We believe that the medium approach is a suitable solution to deal with the AAL systems specificities.

3.1 The medium approach

The medium approach as originally described by Cariou in [8] is defined by:

- **Fixed provided services**. The border of the logical medium specifies its roles, which are the points of interaction with its outside and that define the functional properties of the medium.
- **A distributed architecture** for the implementation. As the logical medium should allow distributed collaborations among roles, its internals are implemented as a set of distributed parts, called managers. Managers collaborate to provide the services specified by the roles. However, despite being distributed, the medium is considered as a whole: the distribution of its internals is considered as non-functional.
- **A design method** *a)* where the communication is not specified as a functional requirement, and *b)* consisting in a set of refinements applied successively, where each refinement considers a particular design concern. Considered design concerns includes the specification of the architecture and of the distribution of the resources, such as algorithms, data types, data management strategies...

The final application will be the interconnection of the managers that implement the logical medium services and a set of clients software components that use the roles provided by the logical medium.

To illustrate this concept, we consider an application that provides local news for the elderly people. The news are provided in a raw textual format by a local news provider, and the elderly can access them, e.g., *via* an application running on a set-top box. In that case, the logical medium consists of two roles: the first one is the role for accessing the news, the second one is the role for providing them

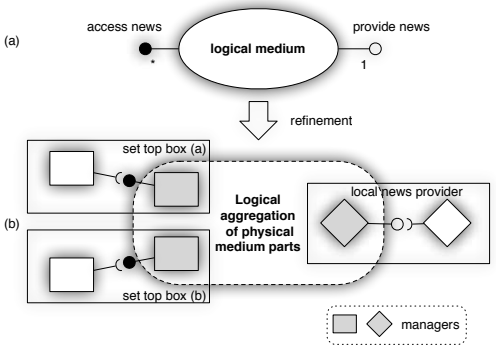

Figure 1: Overview of the design process of a medium (a) a logical medium (b) a refined medium

(fig.1.a). The refinement process is applied to the logical medium until it is fully specified (fig. 1.b).

3.2 A design method to specify variants

Kaboré *et al.* in [16] has automated the design method, using model transformations in a Model-driven architecture (MDA) software design approach [19]. Running successive model transformations refines the logical medium, resulting in an implementation variant. Applying the process several times leads to the design of multiple implementation variants.

The development process of a medium is defined by three successive steps, that must respectively be realized by an expert, a solution designer, and an application designer.

- The **expert** defines a framework composed of a set of metamodels (fig.2). The first one describes the logical medium. Then, each successive metamodel differs from the previous one by the incorporation of a solution for a specific concern. The expert also defines the transitions (e.g., *via* a model transformation language like Kermeta [11]) and the solutions metamodels.
- A **solution designer** designs sets of solution models conforming with the solution metamodels provided by the expert. Each solution model is stored in a solution repository.

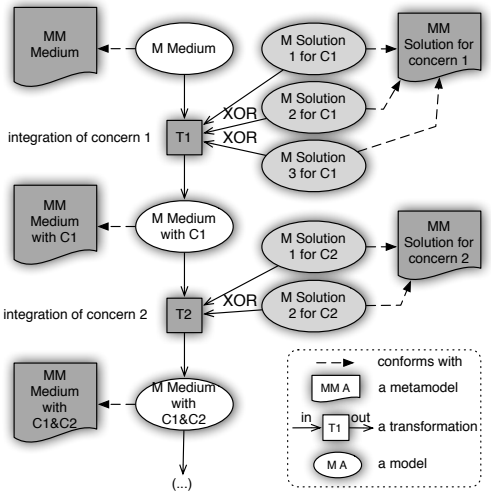

Figure 2: Overview of the refinement process.

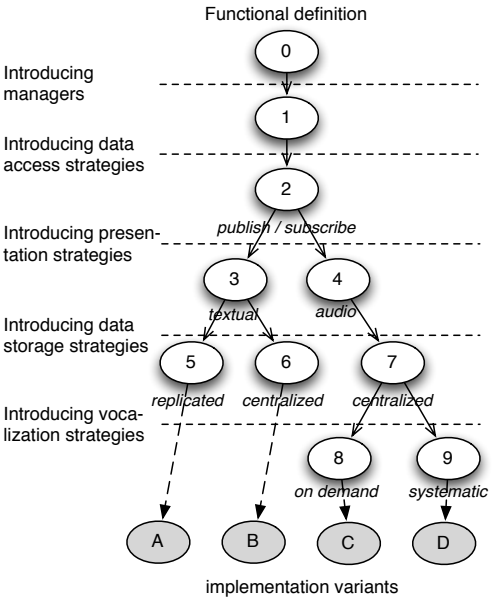

Figure 3: The decision tree for the local news provider example.

- Finally, an **application designer** uses the process designed by the expert, and resolves each concern by choosing a solution from the solution repository. He can also specify multiple alternative solutions for each concern, leading to a specification of multiple implementation variants for a single logical medium, where all variants share the same functional definition.

For more details about the design method, the reader can refer to [21].

3.3 Example

Let's reconsider the previous example about the local news provider. We want to build another version of that medium, that would allow the reading of articles by an avatar instead of a textual presentation. Vocalization is a specialized process that in this case cannot be computed neither by the news provider nor by the news reader due to a lack of resources: it has to be computed by a third element in the distributed application. Also, the vocalization result is a heavy file and the storage capabilities of the news provider and the news reader are inadequate for such files. Considering that the network bandwidth is enough, using a centralized data-storage can be a solution to preserve storage capacities of the news provider and the news reader, and to preserve storage waste due to multiple copies of the same file. Moreover, the vocalization operation could be made using two distinct strategies: *on demand* or *systematically*. The first one vocalizes text-based news just-in-time when an audio-version is requested by a news reader, and is then stored on the data-storage. On the latter, the text-based news are vocalized systematically when provided by the news provider. Both versions have distinct properties: the "*on demand*" one suits a situation where only a few articles are accessed in a vocalized format, the "*systematic*" one fits better if the ratio vocalized/unvocalized grows.

Those modifications imply substantials architectural changes to the text-based news system. The latter may easily be implemented with the data being copied from the publisher to the client using a simple publish / subscribe pattern [13]. With the introduction of the vocalization, the architecture of the distributed application is subject to substantial changes as it gets more elaborate. Using the design method of the medium approach, it is possible to design those multiple variants of the application.

The decision tree represents the decisions made

by the application designer to specify the variants of the logical medium. As presented on figure 3, five concerns have been identified for our example, leading to four variants. The input in the process is a functional definition that specifies the two roles: "access news" and "provide news" (0). The first concern is the manager introduction, which introduces the distributed nature of the software: each role is associated with a manager (1). This is a generic step for every application using the medium approach. Then, a single data access strategy is introduced, a publish/subscribe mechanism where the "access news" role manager queries the medium internals for the data (2). The next concern introduces two data presentation strategies: textual for text-based news (3), and audio for vocalized news (4). In that step, the data formats are being defined. Then, the data storage strategy is introduced. In one case, the data is copied from the provider by the reader (5); in another case, the data is centralized in a specified point, accessed by the readers and fed by the provider (6,7). Finally, the vocalization strategies are introduced (8,9).

That process leads to the definition of four variants of the logical medium (fig. 4). Variants A and B are text-based, variant C and D are audio based. On variant A, the text data are provided by the provider to the reader, and the client keeps a copy in a local storage. On variant B, the only data storage is a centralized point on the software. Variants C and D introduce the vocalization manager, which is in charge of vocalizing submitted text. On variant C, the data is vocalized "*on demand*" upon request of the data storage, whereas on variant D it is done "*systematically*" upon request of the news provider.

3.4 Introduction of the adaptation mechanisms

Phung-Khac *et al.*, in [21], extend this design method by adding adaptation mechanisms in a step called *composition*. The composition merges all the variants and introduces an adaptation framework, DYNACO [4], in the software (fig. 5). The introduction of the adaptation mechanisms allows the dynamic reconfiguration of the application.

The adaptation mechanisms are based on a mapping of the steps of the development process, represented by the decision tree defined by the application designer, in a component based architecture during a step named *modularization*. Using a component architecture allows manipulations on the components, such as component control actions (start, stop), component addition and removal actions (creation, removal, binding, and unbinding), and state and data transfer actions (extract, inject). The adaptation plan for adapting from the replaced to the replacement variant is deduced from the decision tree, and consists in a set of those actions. The data (and state) transfer actions are modeled using a data transfer model. The model is introduced by the application designer once all the variants are generated. The actions includes reading the state and writing it to the new variant.

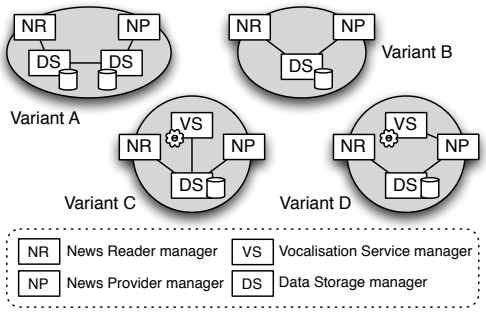

Figure 4: The resulting variants.

Let's extend the previous example. The system includes a mechanism that can detect whether an elderly takes a too long average time to read the articles that are provided. In this case, the application is able to self-reconfigure from the text-based version to the vocalized-based version. Also, another detection mechanism triggers the reconfiguration from the "*on demand*" version to the "*systematic*" version, when the ratio of vocalized articles reaches a predefined number[2].

4 Improvements of the medium development process to fit better AAL systems

The medium approach is generic, and we plan to introduce new features that will make it fit better for distributed AAL systems. We identified the follow-

[2]We voluntarily don't consider whether the triggering of the adaptation is manual or automatic, as it is out of scope of this work

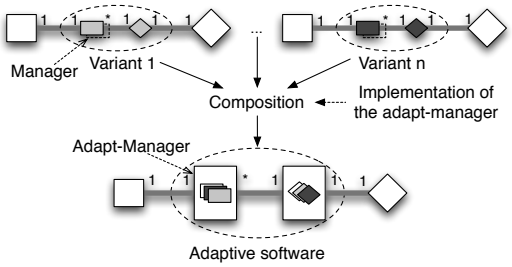

Figure 5: The composition process.

ing limits and potential improvements related to the design method :

Management of heterogeneous target environments. The use of a PIM (Platform Independent Model) in the MDA design method leads to an abstraction of the target platform. It should be possible to specify a new variability by introducing different target platforms from this abstraction, leading to an heterogeneous system. In the context of the AAL system considered in the examples, it should be possible to "move" the reader role from a set-top box to another device such as a hand-held tablet computer, and to transfer data and state between them.

Extending the adaptation beyond the software architecture. The distribution of the medium internals introduces of a notion of cardinality among the components of the assembly. The distinction of "*components assembly*" and "*models of components assemblies*" imposes cardinalities in the latter. A "components assembly" consists in the representation of *instances* of components and their relations, whereas a "model of components assemblies" are made up of representations of *classes* of components with cardinalities.

Considering an adaptive application built on models of component assembly instead of components assembly, it is possible to infer new adaptation possibilities, e.g., to have two variants running simultaneously, by "merging" two models of component assemblies. That can raise interesting properties such as the manageability of disconnections of some instances during an adaptation, and challenges such as the definition of mechanisms to switch back to an homogeneous architecture.

Abstraction of the adaptation mechanism. The use of an adaptation mechanism based on components has consequences on the nature of the transformations during the design process. As the target platforms will became heterogeneous, the adaptation mechanism must take into account the adaptation capabilities of those target platforms, that can differ from component based platforms which rely on reconfiguration of component assemblies: dynamic weaving of aspects, hot updating of code, mobile software agents or even lower level ones such as a basic firmware updates. We plan to provide a model that will be able to handle heterogeneous target platforms by using a refineable abstraction of the adaptation mechanism and of the dependancies on the design process.

A better specification of the state transfer Using the current process, the specification of the transfer of the state and the data from the replaced to the replacement variant during an adaptation is done by an extraction/injection mechanism of data that conforms to a state transfer model. However, some data adaptations are impossible, and that situation is neither explicitly nor implicitly handled. E.g., a variant stores some data using just a footprint of the data computed by a one-way-function. A common example is the use of a sha1 footprint to store passwords. In this case, it is by definition impossible to retrieve the original abstracted data from the replaced variant (*aka* retrieve the original passwords) in order to reinject it to the new version. Those cases have to be tackled using a specific data transfer method that must be explicitly refineable during the design method, to manage the necessary data transformations and the specification of specific behaviors for the cases where the transformed data is incomplete (e.g. to retrieve it for somewhere else).

5 Related work

There are lots of works in the domain of extending the information society by providing services for the elderly people. However, few of them consider customization or adaptation of the provided services.

In the context of the **BelAmI project** [5], Schneider and Becker study solutions for adaptive component based applications in the domain of AAL systems [24]. Their solution aims at resolving

five adaptation scenarios: local adaptation, remote adaptation, conflict negotiation, set point adaptation, and manual adjustment. They define a component architecture involving a *Configurator*, a *ContextManager* and an *AdaptationManager*, respectively to specify configurable components, to get information from the context, and to plan adaptation actions. Also in the BelAmI project, Anastasopoulos *et al.* propose a service oriented middleware for service reconfiguration and dynamic integration: DoAmI (Domain-Specific Ambient Intelligence middleware platform) [1]. Their solution is based on component assemblies, where each component has different configurations (versions). However, in both cases, the adaptation is limited to a reconfiguration of the components or of their layout.

The approach used by Cetina *et al.* in [9] is similar to ours. They design self-adaptive pervasive systems, with an application for smart homes, as SPL (Software Product Lines) using MDA techniques. The system resources are modeled in a *PervML model* (Services, resources as Binding Providers, communication channels as Interactions, Triggers), and a *specific feature model* describes the functionalities to support each user intentions. Then, a *realization model* establishes relations between the PervML model and the feature Model, to describe how goals could be realized. Those models, described as "Scope, Commonality, and Variability" models, are transformed by model transformations to generate a pervasive system. A *variability model* is also generated, for being used by an *Autonomic Reconfiguration Component* in charge of reconfiguring the application, in order to adapt to the addition/removal of goals (services) and resources. This approach is an interesting view of the problem, however the variability is limited to services and resources. Our approach aims at defining a more general variability.

The context of home automation is rich of interesting approaches in adaptive pervasive systems, and some of those concepts could be reused in the context of AAL systems. Hamaoui *et al.* propose an original solution for the adaptation of a pervasive system to its environment [15] based on a combined use of agents and components. The application is able to react to the environment, including the application structure, to provide tailored services and to generate an adapted GUI. Cheung-Foo-Wo proposes the concept of *aspects of assembly* and a domain specific language to specify them, *ISL4Wcomp* [10]. Aspects of assembly express the scenarios of the application, and allows the generation of components and their assemblies. This demonstrate that the combined use of multiple adaptation paradigms could lead to powerful adaptive systems.

Some other works address specifically the heterogeneity problem. Dufrene *et al.* proposes an ADL as a solution for distributed heterogeneous components architectures [12]. Bottaro *et al.* provides a methodology to facilitate the composition of services offered by electronic devices in a pervasive environment [3]. His solution, *HomeSOA*, uses an OSGi extension, the *refined driver*, to resolve the protocols heterogeneity, and the environment dynamicity. Nain *et al.* proposes an application of the schizophrenic middleware to resolve the protocol heterogeneity [17]. However, the heterogeneity considered in those works does not include the reconfiguration capabilities.

In MADAM [14] and its follow-up project called MUSIC [23], the authors proposed an approach for developing adaptive software. The approach is based on *1)* an adaptation conceptual model describing elements concerning adaptation and *2)* an application reference architecture allowing to specify applications as composition plans of components. In the approach, the development of a software system can be realized by using SPL techniques that enable to identify different variation points. Each point indicates several component variants of a composition plan's component type. The developed software system is then executed and controlled by the MADAM/MUSIC framework that allows to reconfigure the system at runtime by selecting running component variant. By the composition plans, the approach supports a large class of adaptive software. However, the correctness of the software depends on developers. Moreover, component variants, that are provided by developers, must implement reconfiguration interfaces that allow to transfer their states during adaptations. This complex task requires much efforts of the developers.

In ADAM [22], Pessemier *et al.* presented an approach to support evolution (and adaptation) of software systems using components and aspects. Im-

plementation of an aspect is modularized and represented as an *aspect component* that can be connected to traditional components through *aspect bindings*. These concepts are supported by a component platform called Fractal Aspect Component (FAC). Thanks to facilities provided by the platform, software systems' evolution can be realized by modifying components, including aspect ones. While this approach focuses on merging the component and the aspect approaches in order to support adaptation, our approach takes into account the inherent heterogeneity of the architecture (some internal parts may be based on aspects, others on components) from the abstraction level (logical medium) to implementation level.

6 Conclusion

The main contribution of this article is the use of a specific design method, based on the concept of mediums, to specify an adaptive architecture for AAL systems, and the identification of the potential improvements that can be done to this process in order to make it more fitted to that specific domain.

We plan to extend and refine the design method to enable some of those identified potential improvements, especially by refining and extending the modeling of transformations. The final goal is to specify an advanced software design methodology for adaptive heterogeneous AAL systems, and to use it to develop a prototype adaptive application.

We plan to test and validate our design methodology for adaptive applications in the Experiment'AAL laboratory, which is a recreated flat for elderly deployed at Télécom Bretagne by the SID team (Innovative Services for Dependent people). It is fully equipped with devices and captors.

7 Acknowledgement

The authors would like to thank the *Direction Générale des Entreprises* for their funding, and the members of the SIGAAL project [26].

References

[1] M. Anastasopoulos, H. Klus, J. Koch, D. Niebuhr, E. Werkman, "*DoAmI - A Middleware Platform facilitating (Re-) configuration in Ubiquitous Systems,*" System Support for Ubiquitous Computing Workshop. 8th Annual Conference on Ubiquitous Computing (Ubicomp'06), 2006.

[2] A. Borsch-Supan, H. Jürges, "*The survey of health, ageing and retirement in europe - methodology.*," Technical report, Mannheim Research Institute for the Economics of Aging, 2005.

[3] A. Bottaro, A. Gérodolle, "*Home SOA : Facing Protocol Heterogeneity in pervasive Applications*," Proceedings of the 5th international conference on Pervasive services, 2008, pp. 73-80.

[4] J. Buisson, "*Adaptation dynamique de programmes et composants parallèles*," PhD thesis, INSA de Rennes, IRISA, 2006.

[5] The BelAmI project, *http://www.belami-project.org/*, last visited on April 2010.

[6] N. Bencomo, P. Sawyer, G. Blair, and P. Grace, "*Dynamically Adaptive Systems are Product Lines too : Using Model-Driven Techniques to Capture Dynamic Variability of Adaptive Systems*," 2nd International Workshop on Dynamic Software Product Lines, Limerick, Ireland, 2008.

[7] N. Bencomo, P. Grace, C. Flores, D. Hughes, and G. Blair, "*Genie: Supporting the model driven development of reflective, component-based adaptive systems*," ICSE 2008 - Research Demonstrations Track, 2008.

[8] E. Cariou, A. Beugnard, and J. Jézéquel, "*An architecture and a process for implementing distributed collaborations*," International Enterprise Distributed Object Computing, pp. 132-143, 2002.

[9] C. Cetina, J. Fons, V. Pelechano, "*Applying Software Product Lines to Build Autonomic Pervasive Systems,*" 12th International Software Product Line Conference, pp. 117–126, 2008.

[10] D. Cheung-Foo-Wo, "*Adaptation dynamique par tissage d'aspects d'assemblage*," p. 222, 2009.

[11] Z. Drey, C. Faucher, F. Fleurey, V. Mahé, D. Vojtisek, "*Kermeta language reference manual*," 2006.

[12] G. Dufrène, L. Seinturier, "*Un ADL pour les Architectures Distribuées Composants Hétérogènes*," 2ème Conférence Francophone sur les Architectures Logicielles, Montréal, Canada, 2008.

[13] P.T. Eugster, P.A. Felber, R. Guerraoui, A. Kermarrec, "*The many faces of publish/subscribe*," ACM Computing Surveys, vol. 35, pp. 114-131, 2003.

[14] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, E. Gjorven, "*Using Architecture Models for Runtime Adaptability,*" IEEE Software, vol. 23, pp. 62–70, 2006.

[15] F. Hamoui, M. Huchard, C. Urtado, S. Vauttier, "*Specification of a Component-based Domotic System to Support User-Defined Scenarios*," Proc. of 21st Int. Conf. SEKE, Boston, MA, USA, pp. 597ï¿½602, 2009.

[16] E. Kaboré, A. Beugnard, "*Implementing a Data Distribution Variant with a Metamodel, Some Models and a Transformation*," In Procs. 8th IFIP DAIS, Oslo, Norway, 2008.

[17] G. Nain, E. Daubert, O. Barais, J. Jézéquel, "*Using MDE to build a schizophrenic middleware for home/building automation*," In ServiceWave 08 : NESSI Conference, Madrid, Spain : Springer, 2008, p. 61.

[18] J. Nehmer, A. Karshmer, M. Becker, and R. Lamm, "*Living Assistance Systems - An Ambient Intelligence Approach* -," International Conference on Software Engineering, 2006.

[19] Model Driven Architecture, *http://www.omg.org/mda*, Object Management Group, last visited on March 2010.

[20] Observatoire des Retraites, "*Face à l'Octoboom, quels Accompagnements ?*", La lettre de l'observatoire des retraites, number 15, December 2007.

[21] A. Phung-Khac, A. Beugnard, J.M. Gilliot, M.T. Segarra. "*Model Driven Development of Component based Adaptive Distributed Applications*," In Procs. 23rd Annual ACM SAC, Fortaleza, Bresil, 2008.

[22] N. Pessemier, L. Seinturier, L. Duchien, T. Coupaye, "*A Component-based and Aspect-Oriented Model for Software Evolution,*" Int. Journal of Computer Applications in Technology, vol. 31, pp. 94–105, 2008.

[23] R. Rouvoy, F. Eliassen, J. Floch, S. Hallsteinsen, E. Stav, "*Composing components and services using a planning-based adaptation middleware,*" Lecture Notes in Computer Science, vol. 4954, pp. 52-67, 2008.

[24] D. Schneider, M. Becker, "*Runtime Models for Self-Adaptation in the Ambient Assisted Living Domain*," 2008.

[25] G. Sundstrom, E. Fransson, B. Malmberg, A. Davey, "*Loneliness among older Europeans*," European Journal of Ageing, vol. 6, pp. 267-275, 2009.

[26] Special Interest Group on Ambient Assisted Living, *http://www.sigaal.org*, last visited on April 2010.

[27] A. Thépaut, M. Segarra, C. Lohr, P. Chapon, "*Adaptation and customization of services for the elderly*," International Society for Gerontechnology, 7th World Conference, Vancouver, Canada, 2010.

[28] J. Gaymu, P. Ekamper, G. Beets, "*Qui prendra en charge les Européens âgés dépendants en 2030 ?*," Population 62, vol. 4, pp. 789-822, 2007.