Pers Ubiquit Comput (2011) 15:353–363
DOI 10.1007/s00779-010-0356-y

ORIGINAL ARTICLE

A design process enabling adaptation in pervasive heterogeneous contexts

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

Received: 29 September 2010 / Accepted: 4 December 2010 / Published online: 19 January 2011

Abstract In the next decades, the growth in population aging will cause important problems to most industrialized countries. To tackle this issue, Ambient Assistive Living (AAL) systems can reinforce the well-being of elderly people, by providing emergency, autonomy enhancement, and comfort services. These services will postpone the need of a medicalized environment and will allow the elderly to stay longer at home. However, each elderly has specific needs and a deployment environment of such services is likely unique. Furthermore, the needs evolve over time, and so does the deployment environment of the system. In this paper, we propose the use of a model-based development method, the *adaptive medium approach*, to enable dynamic adaptation of AAL systems. We also propose improvements to make it more suited to the AAL domain, such as considering heterogeneity and a composition model. The paper includes an evaluation of the prototype implementing the approach, and a comparison with related work.

Keywords Dynamic adaptation · Model-driven engineering · Heterogeneity · AAL · Adaptive medium approach

J.-B. Lézoray (✉) · M.-T. Segarra · A. Phung-Khac · A. Thépaut · J.-M. Gilliot · A. Beugnard
Département Informatique, Technopôle Brest-Iroise, CS83818, Institut Télécom, Télécom Bretagne, 29238 Brest cedex 3, France
e-mail: jb.lezoray@telecom-bretagne.eu

M.-T. Segarra
e-mail: mt.segarra@telecom-bretagne.eu

A. Phung-Khac
e-mail: an.phungkhac@telecom-bretagne.eu

A. Thépaut
e-mail: andre.thepaut@telecom-bretagne.eu

J.-M. Gilliot
e-mail: jm.gilliot@telecom-bretagne.eu

A. Beugnard
e-mail: antoine.beugnard@telecom-bretagne.eu

1 Introduction

In the next decades, most industrialized countries will face important problems with the growth in population aging. Statistics data provide a clear picture of the problem dimension. According to results of the SHARE survey[1] in 2050 the share of the above sixty-five age group will be around 28% instead of 16% nowadays. Specialized institutions will not be able to handle that situation without alternative solutions. 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.

Also, we notice that most AAL applications are either too general or too specific. Indeed, each elderly has specific diseases, capabilities, and habits, and the deployment environment (e.g., the set of available devices to execute AAL services) may be different from one instance to another. Therefore, proposing a too general or too specific AAL application is not a satisfying solution as it may lead to (a) a rejection of the application by the final user, or (b) an inadequate deployment of the application. Similarly, the needs of the elderly may evolve over time (e.g., between night and day, or between winter and summer),

[1] Survey of Health, Ageing and Retirement in Europe, http://www.share-project.org.

and so does the deployment environment of an AAL application (e.g., due to faults or updates of devices). As those evolutions may occur during application execution, AAL applications should have the ability to dynamically adapt to them.

On the other hand, AAL applications are often developed as pervasive systems, in order to provide services independently of user and service location. Then, as a pervasive system, an AAL application is distributed and should be able to execute in an heterogeneous environment, which includes different target platforms. However, few works do compose with heterogeneity, dynamic adaptation, and distribution, despite that each one has strong consequences on the others.

In [13, 16], we proposed a model-based methodology that facilitates the development of adaptive and distributed applications: the *adaptive medium approach*. This paper presents improvements of this approach that raises new interesting capabilities for developing AAL applications related with heterogeneity and adaptability characteristics. We are currently working on applying this methodology to the development of a pervasive indoor AAL application 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 while maintaining social links with their relatives and their circle of friends.

The paper is organized as follows. The next section analyzes the specific needs of indoor AAL systems for elderly people, and introduces the basic functionalities that should be implemented to build them. Section 3 presents the background used for our solution, which is the generic concept of the adaptive medium approach, and its underlying design process based on model-driven engineering (MDE) techniques. The fourth section studies how to extend that method to make it suitable for specific needs of AAL systems, especially by introducing heterogeneity management, and a model for composing mediums. Section 5 presents our current implementation of the adaptive medium approach, and how we plan to extend it to support heterogeneity and composition. Finally, Sects. 6 and 7 present our work regarding research results on AAL systems, heterogeneity, and adaptation, and discusses future work, respectively.

2 Adaptation of AAL systems for the elderly

Ambient Assisted Living systems denote the use of computing technologies to improve users well-being. Services provided by indoor AAL systems can be classified in three subdomains [15]: (a) *emergency treatment services* predict, detect, and/or prevent unsafe situations by propagating alerts in emergency situations (e.g., sudden falls, heart attacks), (b) *autonomy enhancement services* provide tools to postpone the need of assistance by health caregivers or relatives (e.g., a cooking assistance system), and (c) *comfort services* greatly enhance the quality of life (e.g., logistic services, infotainment services, or services to maintain or create social links).

To develop such a system, users and deployment environments specificities that should be taken into account include:

- *Users may have 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. The person with muscular problems must have services with specific simplified control interfaces. Therefore, the AAL system must be customizable, depending on the capabilities of final users.
- *The acceptability of some services may be different*: to be efficient, the system has to be perceived positively by the elderly people and by the circle of people in charge of them (relatives, medical staff,…). For some people, having a camera analyzing their moves and position is inappropriate due to privacy issues, even for a very useful service such as fall detection. For others that type of service is not problematic. Therefore, the AAL system must be customizable depending on the acceptability of final users.
- *Target platforms*: different target platforms are available for an AAL application, e.g., tablets PCs, home automation captors and actuators, iPads, set-top-boxes[2]. Each of them has different capabilities in terms of memory, computing,… Therefore, the AAL system must be customizable depending on the capabilities of all the available target platforms.
- *Home configuration*: number of rooms, network configuration, layout of the devices,… Therefore, the AAL system should be customizable depending on the current configuration of devices at home.

In order to be able to manage all these specificities, we claim that AAL systems for the elderly must be *customizable, distributed*, and *heterogeneous*. They should be customizable in terms of the provided services and target platforms to fit each particular need and deployment environment. They should be distributed by providing different, distributed access points (for medical staff, for the elderly, for the relatives) and having services distributed among available devices, for security and quality of

[2] A set-top box is a programmable hardware connected to a TV set.

service reasons. Finally, the heterogeneous nature of AAL applications is related to the distribution of their services. As these services may run on different target platforms, the capabilities of each must be taken into account.

On the other hand, user needs and application deployment environments evolve over time:

- Some services must be adapted to the evolution of the elderly needs, e.g., between night and day, winter and summer, or in the case of a progressive disease such as Alzheimer. Therefore, the AAL system should change its behavior depending on changes on user capabilities.
- The deployment environment is not static, and available resources may evolve over time, e.g., by upgrades or due to faults. The AAL system should adapt to manage these changes.

In order to manage these evolutions, we claim that AAL applications should be developed as *adaptive software*. Adaptation may be related to the evolution of the elderly needs. The application should then 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). When being related to changes on the deployment environment, the application should be able to adapt for a long time scale (e.g., device availability, network charge balancing,…).

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 the application, considering the common parts and the variation points. As a distributed application is composed of multiple internal parts that collaborate to provide a service, the design and distribution of the internals of the variants is a hard task. The second one is the *specification of the transitions* between the variants of the application, leading to an adaptation plan created either by computation or by specification of transition assets. This is also a hard task as each transition comprises multiple modifications that could be architectural, configurational, and/or parametrical. Moreover, the data 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.

In [13, 16], we proposed a methodology to build such an adaptive application: the *adaptive medium approach*. It is based on a development process, which allows the specification of multiple variants of a distributed software [13] by providing a framework where the software is developed by applying a set of successive refinements based on a model-driven architecture (MDA) design approach[3]. Then, those variants are composed with adaptation mechanisms in order to enable architecture-based runtime adaptation [16]. We believe that our approach is a suitable solution to deal with AAL systems. This section aims at presenting the adaptive medium approach.

3.1 The medium approach

The medium approach as originally presented by Cariou in [4] 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 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: algorithms, data types, data management strategies,…

The final application is the interconnection of the managers that implement the logical medium services, and a set of 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 (Fig. 1a). The refinement process is applied to the logical medium until it is fully specified (Fig. 1b).

3.2 A design method to specify variants

Kaboré et al. in [13] has automated the design method, using model transformations in an MDA software design approach.

[3] http://www.omg.org/mda.

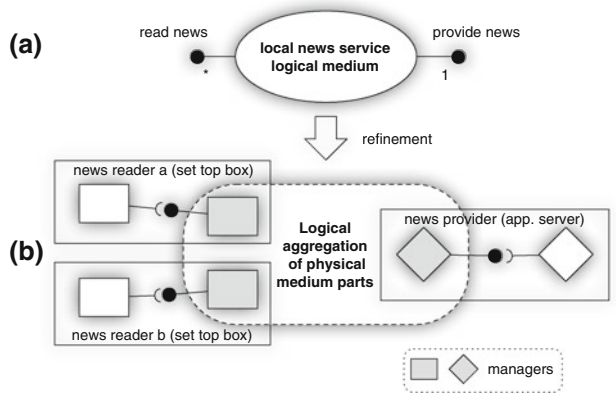

Fig. 1 Overview of the design process of a medium **a** a logical medium **b** a refined medium

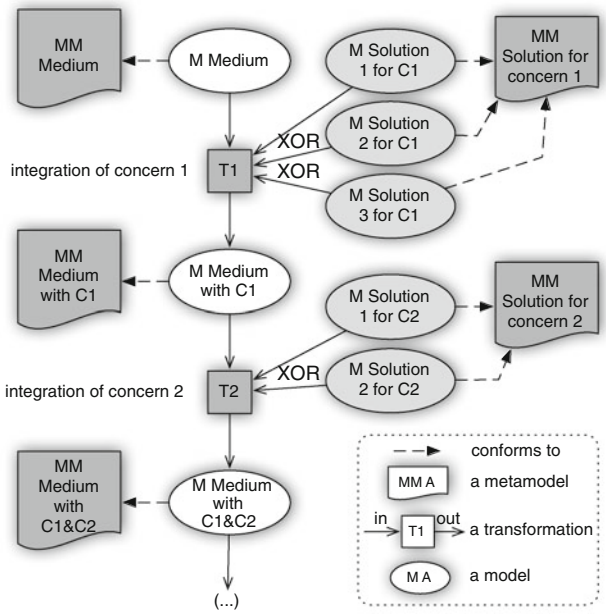

Fig. 2 Overview of the refinement process

Running successive model transformations refines the logical medium, resulting in one implementation variant. The development process of a medium is defined by three successive steps, that must be realized, respectively, by a domain expert, a solution designer, and an application designer. Figure 2 presents two generic refinement steps, including the metamodels, the models, and the transformations.

– The *domain expert* defines a framework composed of a set of metamodels. The first metamodel describes the logical medium, then, each successive metamodel differs from the previous one by the addition of the necessary concepts to introduce a solution for a specific concern. The expert also defines the transitions (via a model transformation language) and the solutions metamodels. In the figure, the elements defined by the domain expert are colored on dark gray.
– A *solution designer* designs sets of solution models conforming with the solution metamodels provided by the expert. Each solution model is then stored in a solution repository. In the figure, the elements defined by the solution designer are colored on light gray.
– Finally, an *application designer* uses the process designed by the expert and resolves each concern by choosing a solution for it from the solution repository. In the figure, the elements defined by the application designer are colored on white.

Using this design process leads to the definition of the implementation of a medium variant that conforms to the initial specification of the logical medium. However, the design process can also lead to the definition of multiple implementation variants: To do so, the application designer selects multiple alternative solutions for each considered 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 may refer to [13, 16].

3.3 Example

Let's reconsider the previous example about the local news provider. Another version of this medium could be developed that allows the vocalization of articles by an avatar instead of a textual presentation. Vocalization is a resource-consuming 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 *systematic*. 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 when the ratio vocalized/unvocalized grows.

The text-based news system could have been easily implemented with the data being copied from the provider to the reader using a simple publish / subscribe pattern. With the introduction of the vocalization, the architecture

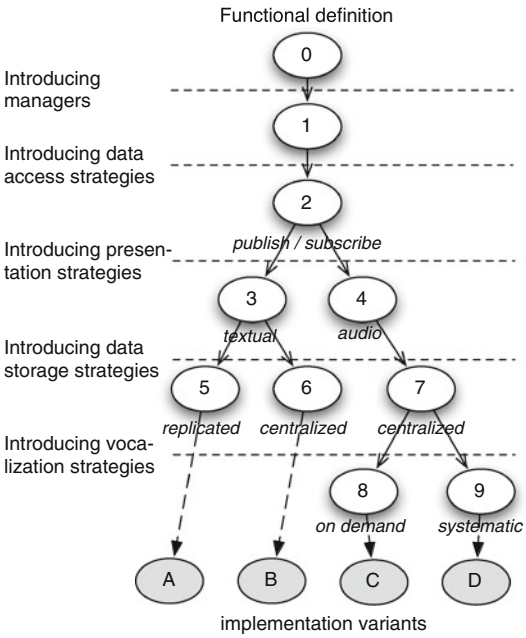

Fig. 3 The decision tree for the local news service example

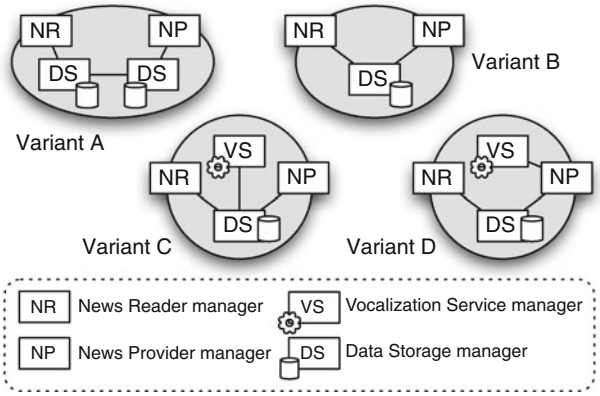

Fig. 4 The resulting variants

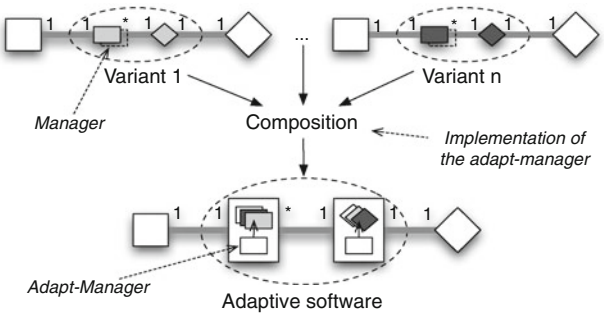

Fig. 5 The composition process

of the distributed application is subject to substantial changes as it gets more elaborate and different variants are possible. Using the medium approach, it is possible to develop these 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 in Fig. 3, five concerns have been identified for our example, leading to four variants. The input of the process is a functional definition of a logical medium that specifies the two roles: "read news" and "provide news" (0). The first concern is the managers 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 "read 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 internal data storage strategy is introduced. In one case, the data are copied from the provider by the reader (5); in another case, the data are centralized in a specified point (a specific manager), accessed by the readers and fed by the provider (6,7). Finally, the vocalization strategies are introduced (8,9).

With those decisions, the process leads to the definition of four variants of the logical medium (Fig. 4), but different decisions could have led to a different number of medium variants. Variants A and B are text-based, variants C and D are audio-based. On variant A, the text data are provided by the provider to the reader who 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 are 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

We extended this design method by adding two new steps. The first one, called *composition* is intended to add adaptation mechanisms [16]. The composition merges all the variants and introduces an adaptation framework, DYNACO [3], in the software (Fig. 5). The introduction of the adaptation mechanisms allows the dynamic reconfiguration of the application.

The second step is named *modularization*. The steps of the development process, represented by the decision tree, are mapped to a component-based architecture. The latter is used by the adaptation mechanisms to manipulate components, such as component control actions (start, stop) and component addition and removal actions (creation,

removal, binding, and unbinding). The adaptation plan for changing from the replaced to the replacement variant consists in a set of actions to manipulate components and to transfer data among them (extract, inject). The first set of actions is deduced from the decision tree. Data transfer actions are deduced from a data transfer model created by the application designer. It fixes which components manage which data and the operations to be used on these components to read and write data.

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 may trigger the reconfiguration from the "*on demand*" version to the "*systematic*" version, when the ratio of vocalized articles reaches a predefined number[4].

4 Improvements of the adaptive medium approach to fit better AAL systems

The adaptive medium approach presented in the previous section facilitates the development of distributed, adaptive applications. However, it fails to provide means to handle heterogeneity and does not consider complex applications, i.e., applications composed of several mediums. In this section, we present the improvements that we plan to introduce in the adaptive medium approach to add both aspects. Section 4.1 presents two kinds of heterogeneity due to the pervasive nature of AAL systems. Then, Sect. 4.2 presents a model of composition of mediums that raises new interesting capabilities for modeling AAL and developing applications.

4.1 Two kinds of heterogeneities

An (adaptive) medium can naturally handle heterogeneity due to its particular architecture composed of multiple layers offering complementary architectural views: the abstract view which specifies the roles, and the managers view which introduces location and distribution of medium parts.

To tackle the heterogeneity of AAL systems, we consider two types of heterogeneity: heterogeneity of target platforms, and intercession capabilities on mediums.

The *specification of the target platforms* constrains the execution context of the software, e.g. programming language, framework, operating system, hardware architecture,… In the adaptive medium approach, this heterogeneity has to be specified at the managers level as it is concerned with location of the medium parts. Many studies provide a link between models and physical target platforms, such as the MDA, Domain-Specific Modeling (DSM), or SCA [20]. However, unlike these approaches, our main goal is to take into account the strong consequences of this heterogeneity on the adaptation framework, and on its composition with the medium.

Table 1 Sample combinations of managers, target platforms, and adaptation framework

Manager	Target platform	Adaptation framework
News reader	OSGi	Life cycle layer
News reader	OSGi	FDF [9]
News reader	Objective/c	ResolveInstanceMethod()
News reader	Fractal	DYNACO [3]
News reader	Fractal	FScript [7]
News reader	Fractal	FDF
News provider	OSGi	Life cycle layer
News provider	Fractal	FScript

The *specification of the Intercession Capabilities on the Medium* (IC_{fm}) specifies how a medium can be interceded. To this end, it specifies the primitives it aims to provide (e.g., move, deploy, weave, connect,…), and that will be used by the adapt-managers to provide an adaptation service over the whole medium. The specification of the IC_{fm} is closely related to the target platform of the medium. Each adapt-manager is composed with a manager whatever its target platform(s). IC_{fm} are provided by an adaptation framework, which is embedded in the adapt-managers. In general, an adaptation framework can only intercede with a specific target platform, although some can intercede with multiple ones. Also, for a single target platform, different adaptation frameworks could be used depending on the type of intercession that is needed (e.g., aspects and component do not offer the same intercession capabilities). Then, the heterogeneity of the target platforms and the need of different types of intercession implies that IC_{fm} are heterogeneous in most cases. In this paper, we do not consider the fact that the adaptation framework itself is linked to a target platform.

Table 1 gives some examples of target platforms and associated adaptation frameworks (that specifies the IC_{fm}). These examples show that the use of a specific adaptation framework is constrained by, but not restricted to the target platform(s) of the medium (e.g., FDF [9] may be used in a Fractal [7] or an OSGi[5] target platform). Moreover, for a given target platform, several adaptation framework may

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

[5] http://www.osgi.org/

be used (e.g., for the Fractal target platform three different adaptation frameworks may be used). Despite this independence of target platforms and adaptation frameworks, the state of the art considers them as a whole. Using that dichotomy goes one step further in separating adaptation mechanisms and functional specification, in particular by introducing the ability to specify multiple solutions for each of them. Moreover, most approaches tackle heterogeneity by introducing an abstraction layer to hide it (see Sect. 6). We state that using such an abstraction layer may lead to the loss of the specificities of each adaptation framework and target platform particularly when the intercession capabilities are weakly compatible with the abstraction layer.

Let's extend our previous example with the considered heterogeneities. The local news medium consists of four managers (Fig. 4), including the news reader manager and the data storage manager. Some of those managers could be deployed on multiple target platforms. For example, news reader manager could be deployed in a set-top-box, to allow the user to read news on the TV set or in an iPad. Therefore, the news reader manager should enforce the API provided by the target platform. When deployed in an iPad, the news reader manager should then be implemented in Objective/C using specific APIs for iOS. When deployed in a set-top-box executing a Java OSGi framework, it should likewise conform to the corresponding API.

To enable adaptation, IC_{fm} must be chosen. Examples of these choices are given in Table 1. In this table, six instances of the news reader manager are available, three of them being deployed in a Fractal component platform, two of them using an OSGi platform, and one executing in a IPad. Among the Fractal-based news reader managers, none of them uses the same IC_{fm}. On the other hand, the second and the sixth news reader manager instances (Fractal and OSGi) uses the same IC_{fm}.

4.2 A model for composing mediums

As stated in Sect. 3, the composition step merges the developed medium variants and introduces an adaptation framework. The composition is a key concept in the specification of the heterogeneities, so to specify them we must consider how the composition modify the composed mediums.

When the adaptation framework is composed with the medium, the result is an adaptive medium. From the managers architectural view, each manager is composed with an adapt-manager (Fig. 5). Like common managers, adapt-managers communicate to provide services, i.e., adaptation mechanisms. Therefore, from the abstract architectural view, they also represent a medium, the

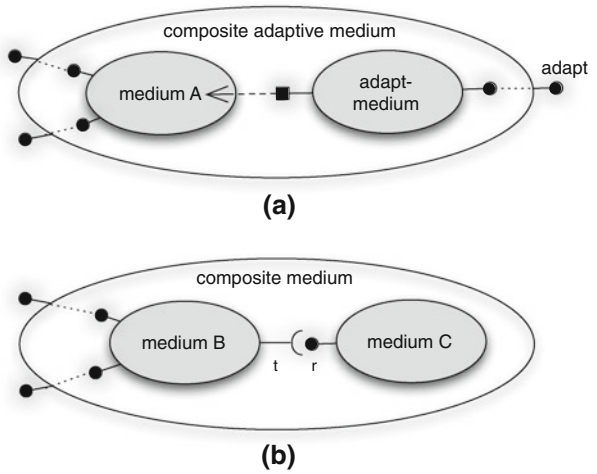

Fig. 6 Detail of the two compositions: **a** a technical composition **b** a functional composition

adapt-medium, as depicted in Fig. 6a. This medium offers two roles: one that aims at being composed with another medium in order to manipulate it, e.g., the composition with *medium A* in the figure, and the second one that provides the "handles" to trigger adaptation, e.g., the role named *adapt* in the figure. Composing a medium with an adapt-medium does not modify the border of the first one. We call this type of composition a *technical composition*. However, technical compositions are not the only ones that can be done. Another type of composition is the *functional composition* (Fig. 6b). It is related with the border of the mediums involved, e.g., role *r* of medium *C* and *t* of medium *B* in the figure. In this case, the role *r* of medium *C* provides a service. The role *t* of medium *B* is then a client of this role, as it requires it. Then role *t* must be specified in a way it is compatible with role *r*. Both the compositions result in a *composite medium*, which shares the characteristics of a primitive medium: its roles are fixed, its architecture is distributed, and communication is considered non-functional.

From the abstract architectural view, the heterogeneities described in the previous section impact either the whole composite medium or only the adapt-medium. In particular, heterogeneity of the target platform impacts the whole composite medium as both the composed and the adapt-medium are concerned by the target platform choice. On the other hand, as the medium to be adaptable may be specified without adaptation aspects, technical composition semantics impacts only the adapt-medium.

Let's redefine our example using this composition model (Fig. 7). In the figure, the composite medium provides three roles, which are the same two roles as previously (*read news* and *provide news*), and another one that triggers adaptations of the whole medium (*adapt*). The roles of the primitive local news medium are promoted to

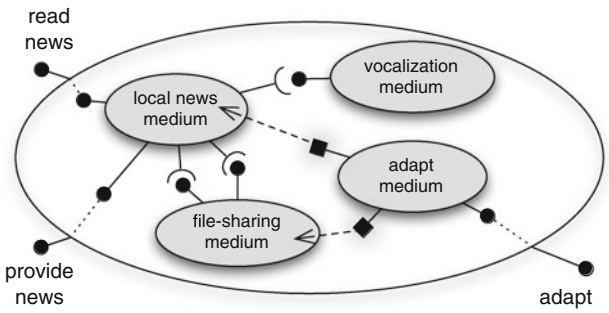

Fig. 7 The news example represented with compositions

the border of the composite medium to reify the two first ones. The last one is provided by an adapt-medium. Two other mediums compose the composite one: the *vocalization medium* which is in charge of vocalizing the textual news, and the *file-sharing medium* which is in charge of data storage. In this model, we assume that the vocalization medium is not adaptive, so it is not composed with the adapt-medium. The composite medium is then the result of three functional compositions and two technical ones.

The approach is convenient for modeling AAL systems, as this high level abstraction is a simple representation for a pervasive system. Indeed, each depicted medium is distributed over the architecture: The representation of such an application with a common component approach would be an awkward task. The approach also promotes the reuse of existing mediums, such as the *file-sharing medium*, which is described in [16].

5 Prototype and evaluation

We have implemented a prototype of an adaptive medium [16]. The prototype is an *adaptive file-sharing medium* based on a generic medium development framework consisting on reusable metamodels, transformations, concerns, and implementations. It consists in a *file-sharing medium*, and an *adapt-medium*, informally composed together to form the adaptive medium. For more information on the implementation, readers may refer to [16].

In Fig. 8, adaptation times of two different adaptations is compared. In *Adaptation 1*, the files storage strategy is changed from a centralized to a peer-to-peer variant with 10 peers involved. In *Adaptation 2*, the file localization data storage strategy changes from a *multiple servers* (2 servers) to a centralized variant. In the figure, adaptation time increases with the amount of managed files. Indeed, both adaptations include the migration of data. In the first case, files are moved from the centralized server and distributed among the 10 peers. In the second case, file localization data should be transferred to the centralized

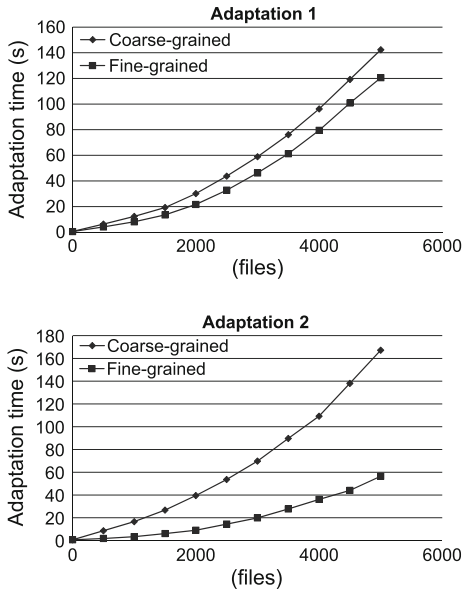

Fig. 8 Adaptation time for two different adaptations

server. The figure also compares coarse-grained and fine-grained adaptation. Coarse-grained adaptation implies changes in all the components of the replaced variant while fine-grained adaptation identify common components between the replaced and replacement variants so that they are kept after adaptation [16]. The results show that both fine-grained adaptations reduce adaptation time. However, adaptation 2 better reduces adaptation time as files are kept and only file localization data (smaller than files) are transferred.

On the other hand, Table 2 compares user-defined and generated software assets in the development of the file-sharing adaptive medium. Thanks to the development framework containing basic metamodels and transformations, software engineers need to define 3 metamodels, 27 models, and 3 transformations in order to generate 128 variants as component diagrams containing 1,536 components. Moreover, the 3 transformations, the 3 metamodels,

Table 2 User-defined and generated elements in the file-sharing adaptive medium development

User-defined	Generated
1 medium model	128 component diagrams
1 variability model	1,536 components
1 process model	
3 solution meta-models[a]	
7 solution models[a]	
3 transformations[a]	
17 decision models	

[a] Reusable for the development of other adaptive mediums

and 7 models are reusable. The number of generated diagrams (128) and components (1,536) are those that developers would need to defined with a classical manual approach to reach identical adaptation time results. This illustrates the advantage of the adaptive medium approach in terms of automating the development.

We plan to reuse, extend, and generalize this prototype to explicitly take into account composition of mediums, heterogeneity of target platforms and intercession capabilities. The prototype will be applied to the design of a system alike the one presented during this paper, significantly improved to match users needs. The prototype will be experimented by the SID research group of Telecom Bretagne by deploying it in the *Experiment'AAL laboratory*, a recreated flat for elderly people, fully equipped with heterogeneous devices and captors [21].

6 Related work

6.1 Adaptation in AAL systems

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 and/or adaptation of the provided services.

In the context of the *BelAmI project*[6], Schneider and Becker study solutions for adaptive component-based applications in the domain of AAL systems [19]. 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, respectively. 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 the reconfiguration of the components or their layout, and the separation of the services and the adaptation mechanism is not clear.

The approach used by Cetina et al. in [5] is similar to ours. Cetina designs self-adaptive pervasive systems, with an application for smart homes, as SPL (Software Product Lines) using MDA techniques. Resources are modeled in a *PervML model*, 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. By using model transformations, a pervasive system is generated as well as a *variability model*. The latter is used to manage addition/removal of services and/or resources when adaptation is needed. Although interesting, this approach limits the considered variability to services and resources. Our approach aims at defining a more general variability, as it also considers non-functional properties of the services.

6.2 Adaptive home automation systems

The context of home automation is also 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 [11] 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. Agents are deployed dynamically, and each agent is capable to manage its internal components, according to the services and the events returned by the equipments of the environment. Then, the GUI is tailored to the actual deployment and state of the application. The approach demonstrates that the combined use of multiple adaptation paradigms could lead to powerful adaptive systems.

Cheung-Foo-Wo proposes the concept of *aspects of assembly* and a domain-specific language to specify them, *ISL4Wcomp* [6]. Aspects of assembly express the scenarios of the application, and allows the generation of components and their assemblies. Aspects of assembly are an interesting layer of reconfiguration over components architecture, which demonstrates that intercession capabilities can be of multiple kind for an architecture: one could manipulated components or aspects of assembly to fulfill an architectural reconfiguration.

6.3 General auto-adaptive systems

In MADAM and its follow-up project called MUSIC [18], 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

[6] http://www.belami-project.org/.

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 [17], Pessemier et al. present an approach to support evolution (and adaptation) of software systems using components and aspects. Implementation 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.

Some work studied adaptation in distributed and pervasive systems, such as Bennour et al. [2] with a dedicated extension of FScript that enables remote interpretation of reconfiguration procedures, and Seinturier et al. [20] with the FraSCAti implementation of SCA. Similarly, Fuentes et al. [10] propose an approach that is similar to our with a component and aspect-based AmI middleware platform named DAOPAmI. They propose to overcome the specification of the communication inside a pervasive application by using aspects to model it, also enabling dynamic evolutions. However, these approaches either do not tackle the heterogeneity of the system, or define a fixed abstraction layer over the distributed components to intercede with them. We consider both as limitative due to the strong relation between the target platforms and the adaptation capabilities.

6.4 Heterogeneity in distributed systems

Some works specifically address the heterogeneity problem. Dufrene et al. proposes an ADL as a solution for distributed heterogeneous components architectures [8]. Nain et al. proposes an application of the schizophrenic middleware to resolve the protocol heterogeneity [14]. Hoareau et al. proposes a middleware where composite components are heterogeneous [12]. However, the heterogeneity considered in those works does not include the reconfiguration capabilities, which are systematically considered as homogeneous.

7 Conclusion

The main contributions of this paper are (1) the use of a specific development method, based on the concept of medium, to specify an adaptive architecture for pervasive AAL systems, and (2) the identification of the improvements of this method to make it more fitted to that specific domain.

Extensions and refinements of the development method include medium composition definition and heterogeneity of target platforms, and intercession capabilities. By defining models for both aspects, an advanced software design methodology for developing adaptive, heterogeneous AAL systems will be proposed.

A first prototype to support the development methodology exists which validates the interest of the method to develop adaptive, distributed applications. Tests and validation of the identified improvements will be performed in the Experiment'AAL laboratory, which is a recreated flat for elderly deployed at Télécom Bretagne. It is fully equipped with heterogeneous devices and captors.

Acknowledgments The authors would like to thank the *Direction Générale des Entreprises* for their funding, and the members of the SIGAAL project [21].

References

1. Anastasopoulos M, Klus H, Koch J, Niebuhr D, Werkman E (2006) DoAmI—a middleware platform facilitating (Re-)configuration in ubiquitous systems. In: System Support for Ubiquitous Computing Workshop. 8th Annual Conf. on Ubiquitous Computing (Ubicomp'06)
2. Bennour B, Henrio L, Rivera M (2009) A reconfiguration framework for distributed components. In: ESEC/FSE workshop on software integration and evolution at runtime, pp 49–56
3. Buisson J (2006) Adaptation dynamique de programmes et composants parallèles. PhD thesis, INSA de Rennes, IRISA
4. Cariou E, Beugnard A, Jézéquel J (2002) An architecture and a process for implementing distributed collaborations. In: Proceedings of the 6th International Enterprise Distributed Object Computing, pp 132–143
5. Cetina C, Giner P, Fons J, Pelechano V (2009) Building self-adaptive services for ambient assisted living. In: Int. Workshop of Ambient Assisted Living (IWAAL), Salamanca, Spain
6. Cheung-Foo-Wo D (2009) Adaptation dynamique par tissage d'aspects d'assemblage. PhD thesis, Univ. de Nice Sophia Antipolis
7. David P-C, Ledoux T, Léger M, Coupaye T (2009) FPath and FScript: language support for navigation and reliable reconfiguration of fractal architectures. In: annals of telecommunications: special issue on software components. The fractal initiative, vol 64, n1/2. Springer, Paris

8. Dufrène G, Seinturier L (2008) Un ADL pour les architectures distribuées composants Hétérogènes. In: 2nd Conf. Francophone sur les Architectures Logicielles, Montréal, Canada
9. Flissi A, Dubus J, Dolet N, Merle P (2008) Deploying on the grid with DeployWare. In: 8th Int. symp. on cluster computing and the grid (CCGRID'08), Lyon, France, pp 177–184, 18–22 May 2008
10. Fuentes L, Jiménez D, Pinto M (2006) Development of ambient intelligence applications using components and aspects. J Univers Comput Sci 12(3):236–251
11. Hamoui F, Huchard M, Urtado C, Vauttier S (2009) Specification of a component-based domotic system to support user-defined scenarios. In: Proc. of 21st int. conf. SEKE, Boston, MA, USA, pp 597–602
12. Hoareau D, Mahéo Y (2008) Middleware support for the deployment of ubiquitous software components. Personal and ubiquitous computing, vol 12. Springer, London, pp 167–178
13. Kaboré E, Beugnard A (2008) Implementing a data distribution variant with a metamodel, some models and a transformation. In: Proc. 8th IFIP DAIS, Oslo, Norway
14. Nain G, Daubert E, Barais O, Jézéquel J (2008) Using MDE to build a schizophrenic middleware for home/building automation. In: ServiceWave 08: NESSI Conference, Madrid. Springer, Spain, p 61
15. Nehmer J, Karshmer A, Becker M, Lamm R (2006) Living assistance systems—an ambient intelligence approach. Int Conf Softw Eng
16. Phung-Khac A (2010) A model-driven feature-based approach to runtime adaptation of distributed software architectures. Phd thesis, Télécom Bretagne
17. Pessemier N, Seinturier L, Duchien L, Coupaye T (2008) A component-based and aspect-oriented model for software evolution. Int J Comput Appl Technol 31:94–105
18. Rouvoy R, Eliassen F, Floch J, Hallsteinsen S, Stav E (2008) Composing components and services using a planning-based adaptation middleware. Lect Notes Comput Sci 4954:52–67
19. Schneider D, Becker M (2008) Runtime models for self-adaptation in the ambient assisted living domain. Fraunhofer Publica, Germany
20. Seinturier L, Merle P, Fournier D, Dolet N, Schiavoni V, Stefani J-B (2009) Reconfigurable SCA applications with the FraSCAti platform. In: International conference on services computing, pp 268–275, Sep 2009
21. Thépaut A, Segarra MT, Lohr C, Chapon PM (2010) Project for the autonomy of the elderly: an approach design for all. In: Int. Soc. for Gerontechnology, 7th World Conf., Vancouver, Canada