Supporting Ambient Assisting Living by using Executable Context-Adaptive Task Models

Estefanía Serral
Department of Decision Sciences and Information Management
KU Leuven, Belgium
Email:estefania.serralasensio@kuleuven.be

Pedro Valderas and Vicente Pelechano
ProS Research Center
Universitat Politècnica de València
Email:{pvalderas, pele}@pros.upv.es

***Abstract*—The amount of elderly people with chronic diseases is constantly increasing, and current health systems are not able to provide a proper supervision. Ambient Assisted Living (AAL) is a new research area that stands for the use of pervasive and mobile technologies in order to increase the quality of life, wellbeing and safety of elderly people. In this work, we present a tool-supported methodology to facilitate the creation of AAL systems through the use of executable models. AAL services are specified by executable context-adaptive task models by using concepts of a high level of abstraction, which facilitate the participation of medical professionals in the AAL specification. The task models are then interpreted and executed at runtime by a software infrastructure that automates the AAL services as specified. Thus, task models are the only implementation of the services, making it easy their further evolution after system deployment. In order to demonstrate the feasibility of our methodology, we have evaluated it in the development of an AAL system for assisting the patients of a nursing home.**

***Index Terms*—Ambient Assisting Living, smart environments, models at runtime, context adaptation;**

I. Introduction

This paper introduces an evolution of the software infrastructure presented in [1] in order to support the development of Ambient Assisting Living (AAL) environments [2]. The ageing of population is making that the number of citizens with chronic diseases is increasing, especially among elderly people. These people require a constant control and supervision that traditional public health systems are not able to provide in a satisfactory way. The advances in the Internet of Things (IoT), and the progress of mobile devices (smart phones, tablets, etc.) and wearables (Google glasses, smart watches, sensors in clothes and body, etc.) allow creating systems that sense patients' context in order to take decisions that adapt system behaviour according to the patients' needs.

This is the main goal of AAL practices, which stand for the use of electronic processes and communication as well as mobile technologies in order to support the practice of medicine and public health. AAL applications include services, products and concepts to increase the quality of life, wellbeing and safety of elderly people. The main goal of AAL is to achieve benefits for the individual (increasing safety and well-being), the economy (higher effectiveness of limited resources) and the society (better living standards) [3]. A clear evidence of the current interest in these topics is the European Ambient Assisted Living joint Program and all the submitted projects [4].

AAL environments require self-adaptive systems to automate AAL services that assist patients when is required in a non-intrusive way. These systems also need to consider the patients and medical staff (which will be the end-users of the system), in the system design in order to properly support medical guidelines and patients demands. Several works such as [5][6][7][8] have worked on the automation of tasks and system adaptation. However, these solutions focus on the technical challenge of adapting a system by analyzing user behavior and environment conditions at runtime, paying little attention to the involvement of end-users from early stages of development, which is crucial on AAL environments.

To improve these challenges, we evolve the software infrastructure presented in [1] to be applied in AAL environments. This infrastructure was proposed for automating tasks in smart homes. This paper extends it to successfully automate assisting services. In addition, we propose a methodology for using this infrastructure in order to support the development of these services from the requirement elicitation until their execution. In order to properly identify the assisting services required by the end-users, we use User Centered Design (UCD) techniques such as Personas and scenarios [9]. The identified services are then described in context-adaptive task models. Task models facilitate the participation of patients and medical professionals in their definition and allow describing the assisting services in a very intuitive manner by using high-level concepts close to the problem domain.

The tasks models are directly interpreted by a software infrastructure at runtime, which executes the described assisting services in the appropriate context. This allows automating the services from the very moment in which task models are defined. Moreover, this facilitates the further evolution of the services if health conditions of patients change: by only updating the models, the services are evolved. With this infrastructure, we can provide patients with a high quality of assistance. In addition, assisting services can be performed in a convenient way for patients since they are analyzed by medical professionals before they are automated by using the task model. Moreover, assistance is self-adaptive according to context, i.e., the software infrastructure reacts and autonomously adapts patient assistance according to each context. In order to evaluate our approach we have used it to develop an AAL system that assists the patients of a Nursing Home.

The rest of the paper is organized as follows: Section II presents the related work. Section III introduce the proposed methodology to support assisting services from their requirements to their execution. Section IV presents the process to capture the assisting service requirements. Section V describes the models proposed to specify the needed assisting services. Section VI presents the software infrastructure that automates assisting services in a context adaptive way and allows their evolution after system deployment. Section VII validates the approach using a case study based evaluation. Section VIII concludes the paper and proposes further work.

II. Related Work

An AAL scenario is characterized by being connected, context-adaptive and anticipative. In order to confront the creation of these environments, AAL systems are usually structured in three levels [10]: Hardware (sensing, wireless networks), Middleware (data capture, data safety, IT integration) and Services (biosignal processing, application-orientated processes, community services).

A lot of research work has been done in all three levels. An overview on assisting hardware technology is given in [11]. It addresses video-monitoring, remote health monitoring, electronic sensors, and equipment such as fall detectors and door monitors. This technology alone is not enough to coordinate hardware components in order to provide complex assisting services. Thus, only panic buttons-based solutions can be provided.

To enable increased safety and wellbeing in a specific environment, it has to become intelligent with the help of pervasive devices, which are capable to register changes in the physical environment and thus actively interact in a process. In addition, a system that coordinates these devices is required. In this context, there are a number of research projects focusing on the development of AAL middlewares. For example, the PERSONA project [12] proposes a middleware platform for the implementation of semantic assisting services. The Aware Home [13] project built up a living lab, in which they tested the users acceptance of technology, building up a bridging framework for universal device interoperability in pervasive systems. The mission of I-Living [14] is developing an assisted-living supportive software infrastructure that allows disparate technologies, software components, and wireless devices to work together. Tasks provided in I-Living are such as activity reminding, health monitoring, personal belonging localization, emergency detection, and so on. However, the available services provided by these three projects are closed and still limited, and they do not provide tools that facilitate end-users to participate in the definitions of these services.

According to [15], there are three types of assisting services: (a) Emergency treatment, which faces situations such as sudden falls, heard attacks, strokes, panics, etc.; (b) Autonomy enhancement, which allows replacing medical and social care personnel by appropriate system support such as a cooking assistance system for people with visual defects; and (c) Comfort, which covers all areas that do not fall into the categories (a) and (b) such as social contact assistance, infotainment assistance, logistic assistance, etc.

Independently from their type, AAL services imply the execution of a set of tasks in a coordinated way. For instance, a service that treats a fall may require analyze patient's location and state, and people surround them, and alert doctors or caregivers if a fall emergency is detected; a service for improving autonomy may require to check the fridge for essential products and decide to make a shopping order if needed according to the diet of the users; and a service that manages comfort may require to graduate the light intensity and the temperature, close or open blinds and windows, and play certain music according to the users' taste.

Note that the above presented AAL approaches provide little support to define this type of coordination, where tasks are executed in a specific order depending on environment conditions, users' state, or the outputs of previous tasks. The research fields of task automation and context-awareness play key roles in order to support these three types of services. Machine-learning approaches have attempted to deal with the automation of user routines by automatically inferring them from past user actions [5][6]. These approaches have done excellent work by automatically learning from user behavior; however, assisting services need to be available from the very beginning deployment of the system and a learning process is not acceptable [16]. In addition, these approaches may be intrusive for users because the repeated execution of an action does not imply that the patients or caregivers wants this automation. Also, they reproduce the actions that users have frequently executed in the past and in the same manner that they were executed. This prevents user tasks from being carried out in a more efficient and convenient way, i.e., including medical professional guidelines, and does not allow tasks that users did not perform before to be automated.

Context-aware rule-based approaches have made great advances in introducing context into software systems. To automate user tasks, they program rules that trigger the sequential execution of actions when a certain context event is produced [7][8]. However, these techniques are only appropriate for automating relatively simple tasks [17]; hence, they usually require large numbers of rules. If we also consider that these rules have to be manually programmed [17], the understanding and maintenance of the system may become very difficult.

In this work, we propose a solution based on executable task models. The concept of task is intuitive enough to be understandable by medical professionals and persons resposible of the patients, facilitating their involvement in the development process; they provide rich expressiveness that allows us to precisely describe the assisting services that the system must support to face specific situations; and they can be automatically interpreted by a software infrastructure from the initial deployment of the system. Historically, task-oriented computing uses task modelling to facilitate the interaction of users with the system. These systems have proven that task modelling is effective in several fields such as user interface modelling [18], assisting end-users in the execution of tasks [19], etc. These works show the growing usage of task modelling and its remarkable results and possibilities to model system behavior. However, none of these works attempt to describe AAL services. Hence, they provide neither enough expressiveness to specify them nor enough accuracy to allow their subsequent automation.

Finally, note that, unlike the above works, our approach makes a step further for providing an integrated approach, covering all stages of the development process from requirement elicitation to service execution. We concretely exploit knowledge gathered by UCD techniques to derive the executable models that represent the AAL services.

III. Methodology to Support AAL

To achieve the automation of AAL services, we propose a model-driven development methodology that supports their development from requirements elicitation to their execution. The methodology is proposed considering the following main aspects:

- AAL services must be automated according to specific medical guidelines and to the requirements provided by medical professionals and patient responsible persons. This is essential so that the tasks that are automated can assist to the patients and medical staff (end-users of the system) in the best way. If this information is not taken into account, the AAL services could be very intrusive for the end-users of the system, bothering them, interfering in their goals, or even being dangerous. For instance, according to medical guidelines, a softer room illumination and relaxing music playing may help to make patients more relaxed. If these guidelines are not considered, medical or security staff would be needed every time an aggressive behaviour is detected. Due to the medical context, and the imprecise and ambiguous nature of patient behaviour, it is very difficult for a system to sense or infer this information [20]. Therefore, **the participation of the corresponding end-users is** absolutely **necessary** for supporting AAL.
- **The AAL services must be context-adaptive**. Context information is essential to be able to execute the assiting tasks in the opportune situation. Therefore, AAL services must be described in a context-adaptive way. For instance, if a health anomaly is detected in a patient when s/he has fallen, the appropriate nurses or doctors should be inmediately notified; however, if the patient health is not in risk, it is enough to notify the nearest available caregivers.
- **The evolution of the AAL services must be facilitated after system deployment**. Some AAL services might never change; however, most of them will. Patients' behaviour and health may change over time and the automated services to support them need to be adapted to these changes. Otherwise, the system may become useless and intrusive. Since these types of changes cannot be anticipated at design time, the automation of AAL services must be performed in such a way that their evolution after system deployment is facilitated at runtime.

In order to deal with these aspects, the methodology we propose consists on the following steps (see Figure 1):

- *Requirements Elicitation: AAL Service Identification.* In order to capture of medical guidelines and to properly capture the end-users requirements, we use UCD tecniques to facilite the participation of the specific end-users. We use UCD techniques because they give extensive attention to needs, wants, and limitations of users at each stage of the development process, which is crucial in the design of AAL systems. This methodology step will be further explained in Section IV.
- *AAL Service Modelling.* This activity consists of modelling the AAL services that must be automated by the system. An AAL service is a set of tasks that are habitually performed in a certain context for assisting patients and medical staff. The following steps must be followed to specify the identified AAL services:
  - Context modelling. Analysts specify the context properties on which the AAL services depend, create the necessary rules to infer properties values, and set the property values that need to be manually introduced.
  - AAL service modelling. Using the context-adaptive task model, the analysts specify the AAL services to be automated according to the context previously specified. Each AAL service consitutes a coordination of tasks and is specified as a task hierarchy in which the service represents the highest task in the hierarchy. Each service needs to be progressively broken down into simpler tasks until they can be automatically executed by a pervasive device.
  - Modelling validation. The AAL services' modelling is validated with the end-users to ensure that the tasks will be automated according to patients' needs and medical guidelines. Thus, following an iterative process, the service modelling (and if needed the context modelling), must be refined with the end-users participation until they agree with the specified AAL services. It is important to note that, in this way, the modelling is complemented by both the knowledge of the system analysts, which contributes to improving the performance of the identified AAL services; and the knowledge of the medical professionals and persons responsible of the patients', which contributes to taking into account the needs and demands of the end-users. After validating the service modelling with the end-users, analysts also validate that the models are correctly formed and without inconsistencies.
  - Device linkage: once the modelling has been validated, the analysts link each leaf task with a perva-

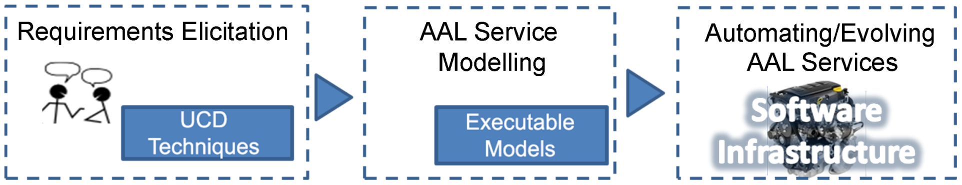


Fig. 1. Methodology for supporting AAL Services

sive device that can carry it out. Pervasive devices can control the objects in the environment (e.g., switching lights on, activating the security alarm, etc.) and sense context information (e.g., detection of presence, measurement of temperature, etc.). Specifically, we consider a pervasive device to be an entity that provides a coherent set of functionality which is described in terms of atomic operations (or methods). Thus, the linkage between leaf tasks and the corresponding pervasive device is made in the task model by indicating the name of the corresponding pervasive device and its operation. The implementation of these devices is out of the scope of this paper; an approach such as PervML [21][22] could be used for developing them.

Since the used context-adaptive task models are executable models [23], this step finishes the AAL service development. This also allows that the specified AAL services can be validated by using prototypes. This would require that the automation of the specified AAL services is done in a simulation mode. This mode should allow analysts to cause context changes and to easily observe the execution of the services according to context.
This methodology step will be further explained in Section III.

- *AAL Service Automation*. To enable the automation of the specified AAL services in the opportune context, the following two steps must be performed:
  - Deployment of the system in the target platform. To deploy the system, analysts install each component of the software infrastructure in an OSGi server. We use an OSGi server [24] because it provides numerous benefits and facilities to make dynamic updates, to easily reuse components, or to deploy the system. In addition, the context model and the task model where the AAL services are specified must be saved in the folder where OSGi is installed.
  - Running the system. To run the system, analysts start the components installed in OSGi. From this moment, the context is continuously monitored and the specified AAL services are automated in the appropriate context. It is important to note that, since the models are directly interpreted, they are the only representation of the AAL services to be automated. This facilitates their understanding, maintenance and evolution.

  This methodology step will be further explained in Section VI.
- *Evolution of the AAL services if needed.* Medical staff and patient behaviour as well as patients' health may change over time and the AAL services that are automated may become obsolete or useless. If this happens, the automated AAL services can be evolved according to the new requirements. To allow this evolution, evolution mechanisms are provided. This methodology step will be further explained in Section VI.

In the next sections we explain how these steps are supported.

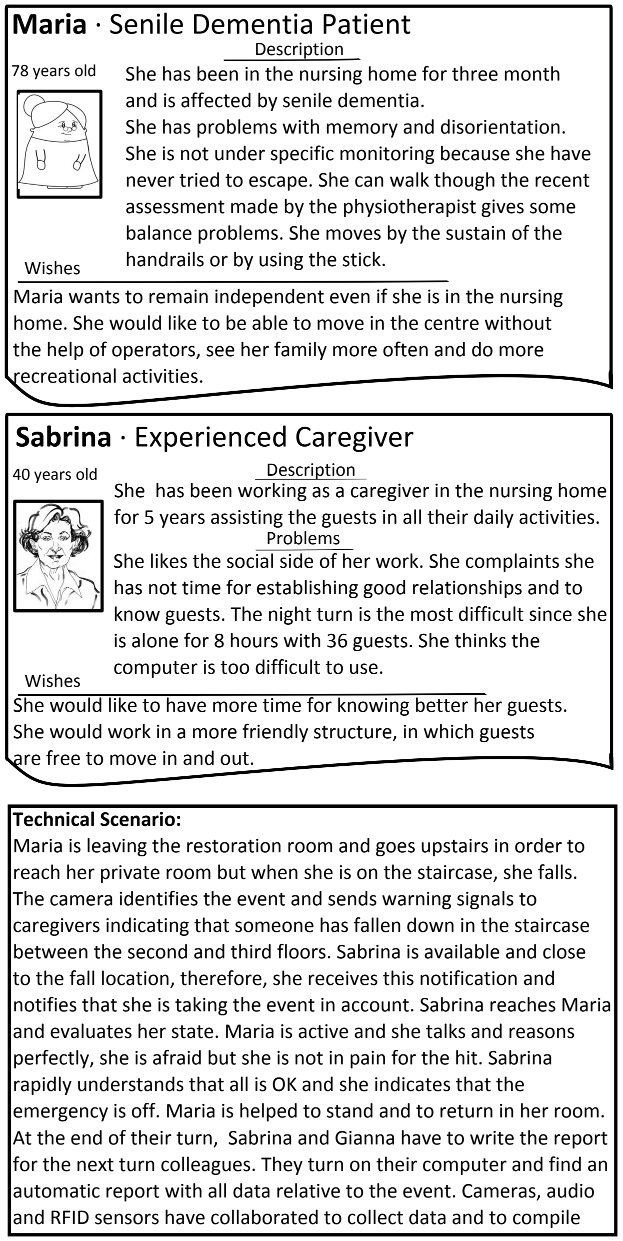
IV. AAL Service Requirement' Elicitation

In order to capture the requirements for automated services, we use UCD techniques. AAL services can be defined for specific patients that suffer from very concrete disabilities or for a group of patients that share a same profile. Therefore, the elicitation process is based on the description of personas and technological scenarios. Their conjunct use increases the ability to identify problems and exceptional cases [25] and to envisage the system to be [26].

Personas are descriptive models of system-to-be users based on behavioural data, derived from patterns observed during interviews, with the aim of representing the diversity of observed motivations, behaviours, and mental models [27]. Examples can be found in Figure 2.

In the context of AAL, a Persona may represent a single patient with specific needs, a group of patients that share same needs, or a medical profile (a doctor, a nurse, a caregiver, etc.). In the case of being either a group of patients or a medical profile, they are personified through a fictitious character that represent them. For the former, the character has the needs that are shared by all of them; for the latter, the character represents the professional skills that are shared by all the people of this profile. This facilitates a shared understanding of who the patients and medical professionals are, and what they need or can provide in order to make decisions about AAL services. In addition, Personas provide a powerful tool for communicating between computer analyst and medical professionals in order to develop and evaluate AAL services. Figure 2 shows two examples of Persona. The first one, Maria, represent a patient with senile dementia; the second one, Sabrina, represent an experienced caregiver.

Personas are complemented with *technological scenarios* that illustrate how they interact with the system. Technological



Maria · Senile Dementia Patient

78 years old

Description

She has been in the nursing home for three month and is affected by senile dementia.
She has problems with memory and disorientation.
She is not under specific monitoring because she have never tried to escape. She can walk though the recent assessment made by the physiotherapist gives some balance problems. She moves by the sustain of the handrails or by using the stick.

Wishes

Maria wants to remain independent even if she is in the nursing home. She would like to be able to move in the centre without the help of operators, see her family more often and do more recreational activities.

Sabrina · Experienced Caregiver

40 years old

Description

She has been working as a caregiver in the nursing home for 5 years assisting the guests in all their daily activities.

Problems

She likes the social side of her work. She complaints she has not time for establishing good relationships and to know guests. The night turn is the most difficult since she is alone for 8 hours with 36 guests. She thinks the computer is too difficult to use.

Wishes

She would like to have more time for knowing better her guests. She would work in a more friendly structure, in which guests are free to move in and out.

Technical Scenario:
Maria is leaving the restoration room and goes upstairs in order to reach her private room but when she is on the staircase, she falls. The camera identifies the event and sends warning signals to caregivers indicating that someone has fallen down in the staircase between the second and third floors. Sabrina is available and close to the fall location, therefore, she receives this notification and notifies that she is taking the event in account. Sabrina reaches Maria and evaluates her state. Maria is active and she talks and reasons perfectly, she is afraid but she is not in pain for the hit. Sabrina rapidly understands that all is OK and she indicates that the emergency is off. Maria is helped to stand and to return in her room. At the end of their turn, Sabrina and Gianna have to write the report for the next turn colleagues. They turn on their computer and find an automatic report with all data relative to the event. Cameras, audio and RFID sensors have collaborated to collect data and to compile

Fig. 2. Example of personas en technical scenario

Scenarios are short narrative stories that represent Personas in their context, supported by the envisaged technology (see Figure 2). Many techniques can be profitable for envisioning system functionalities and services; examples are: internal meetings, brainstorming, focus groups and scenarios. The output of these techniques are the technological scenarios that concretely describe the behaviour of services as experienced by specific, though fictional, users. Stories help the design teams in negotiating a shared representation of the domain and hence a more effective collaborative elicitation of requirements. In this work, scenarios describe how the system interact with patients and medical professionals when specific situations occur. For instance, the scenario presented in the bottom side of Figure 2 explain how the system must act when a patient fall is detected.

V. AAL Service Modelling

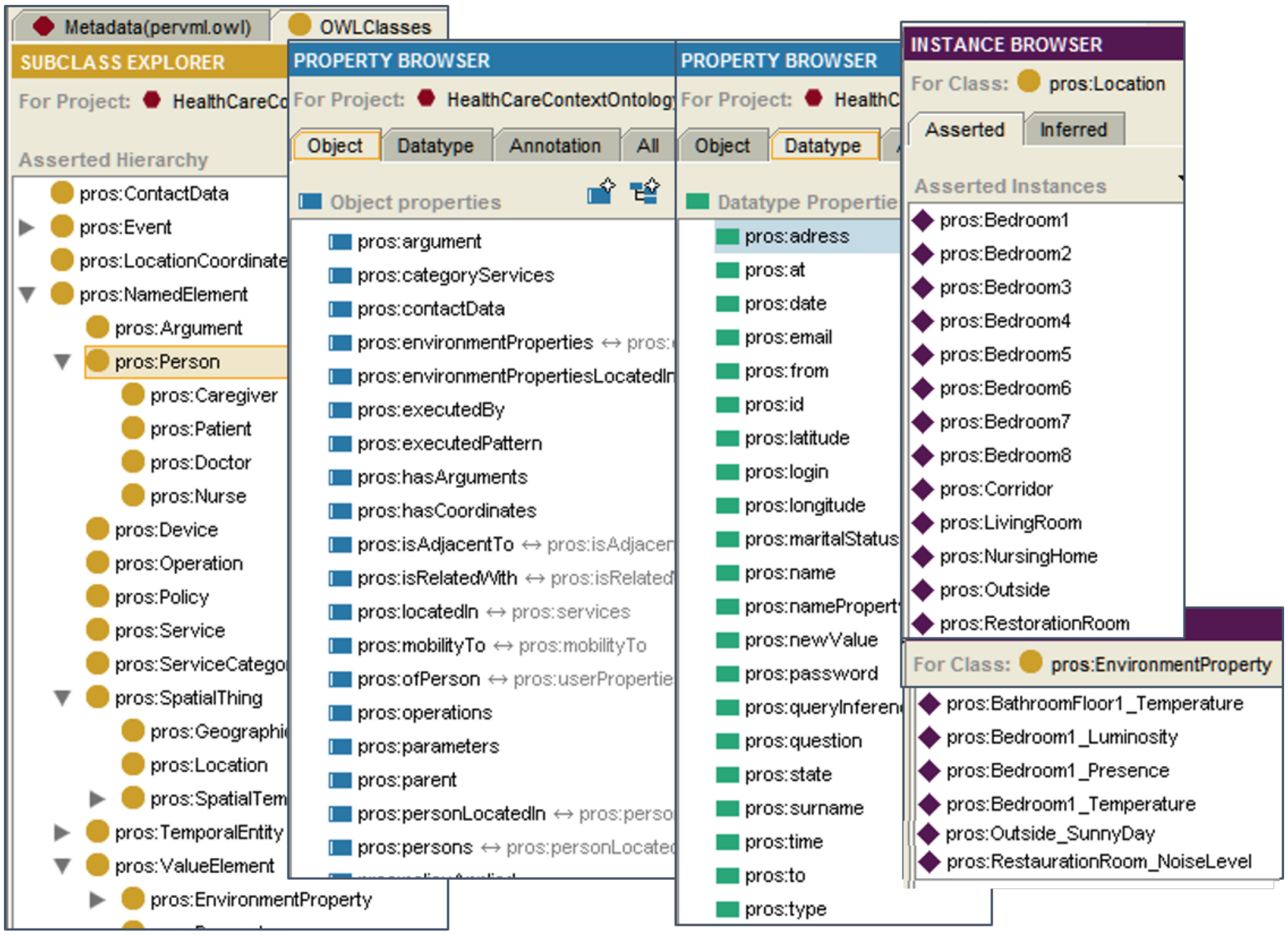
Once personas and scenarios are defined, the AAL services to be automated must be identified and modelled from the obtained requirements. According to the complexity of the system, the modelling can be done manually by directly analysing the requirements, or it could be done by using some type of transformation technique such as the one presented in [28], which is based on the definition of intermediate goals.

The modelling of the AAL services is performed by specifying two models: a context model (which specifies the context on which the services to be automated depend), and a context-adaptive task model (which describes the tasks that must be carried out for each service according to the context described in the context model).

The context model represents the context relevant for the AAL services so they can be executed in a non-intrusive way. Specifically, the model represents information regarding Patients, Medical Staff, Locations, Environment Properties, Policies, Temporal Properties, Services and Events. The model is specified in the Ontology Web Language (OWL)[29], which is an ontology markup language W3C standard that greatly facilitates knowledge automated reasoning and inference. Thus, the classes of the ontology are defined as OWL classes, their relationships as OWL object properties, their attritutes as datatype properties, and the context specific to the system is defined as OWL individuals. For specifying the context model, we use Protégé [30], which is a free open source ontology editor. A context model example created using Protege is shown in Figure 3. From left to right, this model shows some context classes (such as Patient, Caregiver, Location, EnvironmentProperty. etc.), some relationships among these classes (such as locatedIn and isRelatedWith), some data properties of the classes (such as email and id), and some individuals (such as RestorationRoom and RestorationRoom_NoiseLevel). This model contains the following types of context information:

- Manually introduced, such as the personal information of the patients and caregivers. This information is introduced by the analysts.
- Automatically captured, such as locations and health parameters of patients. This information is captured by the context manager that will be explained in the next section.
- Automatically inferred, such as healthAnomaly, which is set to true when a health parameter is outside the normal values. This information is automatically set by inference rules created in the ontology.

Using the context model, the task model describes context-adaptive AAL services precisely and at a high level of abstraction. As an example, Figure 4 shows the modelling of the AAL service that supports the scenario of patient falls. The root task of the hierarchy represents the service and is associated to a context situation, which indicates the context conditions


Metadata(pervml.owl) | OWLClasses

SUBCLASS EXPLORER
For Project: HealthCareCo
Asserted Hierarchy
- pros:ContactData
- pros:Event
- pros:LocationCoordinate
- pros:NamedElement
  - pros:Argument
  - pros:Person
    - pros:Caregiver
    - pros:Patient
    - pros:Doctor
    - pros:Nurse
  - pros:Device
  - pros:Operation
  - pros:Policy
  - pros:Service
  - pros:ServiceCatego
  - pros:SpatialThing
    - pros:Geographi
    - pros:Location
    - pros:SpatialTem
  - pros:TemporalEntity
  - pros:ValueElement
    - pros:EnvironmentProperty

PROPERTY BROWSER
For Project: HealthCareContextOntolog
Object | Datatype | Annotation | All
Object properties
- pros:argument
- pros:categoryServices
- pros:contactData
- pros:environmentProperties ↔ pros:
- pros:environmentPropertiesLocatedIn
- pros:executedBy
- pros:executedPattern
- pros:hasArguments
- pros:hasCoordinates
- pros:isAdjacentTo ↔ pros:isAdjacen
- pros:isRelatedWith ↔ pros:isRelated
- pros:locatedIn ↔ pros:services
- pros:mobilityTo ↔ pros:mobilityTo
- pros:ofPerson ↔ pros:userPropertie
- pros:operations
- pros:parameters
- pros:parent
- pros:personLocatedIn ↔ pros:perso
- pros:persons ↔ pros:personLocated

PROPERTY BROWSER
For Project: HealthC
Object | Datatype
Datatype Propertie
- pros:adress
- pros:at
- pros:date
- pros:email
- pros:from
- pros:id
- pros:latitude
- pros:login
- pros:longitude
- pros:maritalStatus
- pros:name
- pros:nameProperty
- pros:newValue
- pros:password
- pros:queryInferen
- pros:question
- pros:state
- pros:surname
- pros:time
- pros:to
- pros:type

INSTANCE BROWSER
For Class: pros:Location
Asserted | Inferred
Asserted Instances
- pros:Bedroom1
- pros:Bedroom2
- pros:Bedroom3
- pros:Bedroom4
- pros:Bedroom5
- pros:Bedroom6
- pros:Bedroom7
- pros:Bedroom8
- pros:Corridor
- pros:LivingRoom
- pros:NursingHome
- pros:Outside
- pros:RestorationRoom

For Class: pros:EnvironmentProperty
- pros:BathroomFloor1_Temperature
- pros:Bedroom1_Luminosity
- pros:Bedroom1_Presence
- pros:Bedroom1_Temperature
- pros:Outside_SunnyDay
- pros:RestaurationRoom_NoiseLevel

Fig. 3. Context Model Example

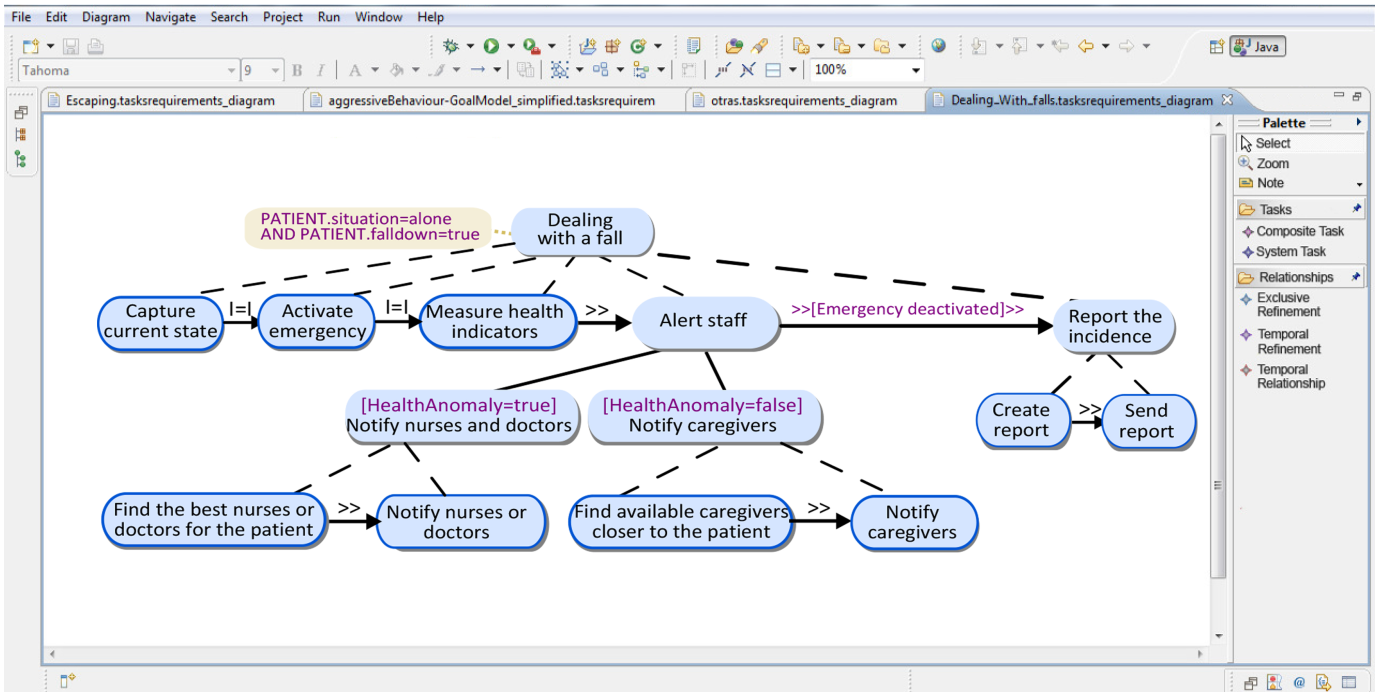

File Edit Diagram Navigate Search Project Run Window Help
Tahoma
9
100%
Java
Escaping.tasksrequirements_diagram
aggressiveBehaviour-GoalModel_simplified.tasksrequirem
otras.tasksrequirements_diagram
Dealing_With_falls.tasksrequirements_diagram
Palette
Select
Zoom
Note
Tasks
Composite Task
System Task
Relationships
Exclusive Refinement
Temporal Refinement
Temporal Relationship
PATIENT.situation=alone AND PATIENT.falldown=true
Dealing with a fall
Capture current state
|=|
Activate emergency
|=|
Measure health indicators
>>
Alert staff
>>[Emergency deactivated]>>
Report the incidence
[HealthAnomaly=true] Notify nurses and doctors
[HealthAnomaly=false] Notify caregivers
Create report
>>
Send report
Find the best nurses or doctors for the patient
>>
Notify nurses or doctors
Find available caregivers closer to the patient
>>
Notify caregivers

Fig. 4. Service modelling using the context-adaptive task model: dealing with a fall AAL service

whose fulfilment starts the execution of the service (*PATIENT.situation=alone AND PATIENT.falldown=true*). The root task is broken down into simpler tasks (*Capture current state, Active emergency, Measure health indicator, Alert staff, Report the incidence*). An intermediate task must be broken down until the leaf tasks can be executed by an available pervasive device. Each leaf task must be related to a pervasive device that can carry out the task. For instance, the active emergency task is associated to a pervasive device that interacts with the emergency system to turn it on. This relation is established by simply indicating the device identifier.

If the tasks of the same parent are related to each other, they are carried out in a sequential order according to the indicated temporal relationships. These relationships may depend on context. Thus, in the example, the current state is captured at the same time (which is indicated by the temporal relationship |=|) that the emergency is activated and the health indicator are measured; after that (temporal relationship >>) the staff is alerted; and finally, the incident is reported when the emergency is deactivated (temporal relationship with a context condition indicated between brackets >>*[]*>>).

In addition, a task can have a context precondition (represented between brackets before its name), which defines the context conditions that must be fulfilled so that the task is performed (e.g., the notify nurses and doctors task is only executed if HealthAnomaly is true). If the tasks of the same parent are not related to each other, only the first task whose context precondition is satisfied is executed. For instance, if the Notify nurses and doctors is executed its sibling tasks Notify caregivers is not.

For user-intensive systems, ontology classes can be used in the context conditions instead of individuals. The condition is satisfied when it is satisfied for one of the individuals of the used class. For instance, the *Dealing with a fall* AAL service has to be executed for every patient. As shown in Figure 4, instead of specifying the same AAL service for each patient, we specified the service once and used the situation and falldown context properties of the PATIENT class in its context situation, indicating by using capital letters that it is an ontology class and not an individual. Thus, the context condition has to be checked for every individual of the Patient class and is activated if any patient fulfils the condition.

For facilitating the specification of the task model, we developed a graphical editor using the Eclipse platform, and the EMF and GMF plugins. By using this editor, the model can be graphically edited as shown in Figure 4. These descriptions are stored in XMI (XML Metadata Interchange), which is machine-interpretable at runtime.

More details about the context model, and the task model and its editor can be found in [23].

VI. AAL Service Automation and Evolution

In order to automate the AAL services as specified in the models and facilitate service evolution after system deployment, we have used the software infrastructure presented in [1]. This infrastructure, which is shown in Figure 5, directly interprets the context model and the task model at runtime to automate the AAL services as described. This infrastructure is built on top of the pervasive devices used to sense context changes and to perform the leaf tasks of the AAL services specified in the models.

The software infrastructure is composed by the following components (see Figure 5): mechanisms for managing the models at runtime, a context manager, and an automation engine.

Mechanisms for managing the models at runtime. In order to manage the context model and the task model at runtime, we have designed and implemented Ontology-based Context model management mechanisms (OCean) and Model-based User Task management mechanisms (MUTate). These mechanisms can be downloaded from http://www.pros.upv.es/art/.

- OCean: The context on which the behaviour patterns depend is specified in the context model as OWL individuals. Thus, in order to manage these individuals, a set of Ontology-based Context model management mechanisms (OCean) is needed. OCean allows, for instance, updating the individuals of the context model, creating a new individual (e.g., the *idealTemperature* individual of the *Preference* ontology class), and reading its properties or modifying them when needed. We have extended OCean to support the checking of context conditions in user-intensive services. In this way, when an ontology class (represented in capital letters) is used in a condition, the condition is considered as satisfied when it is satisfied for one of the individuals of the class.
- MUTate: In order to support the management of the task model, a set of Model-Based User Task management mechanisms (MUTate) is needed. MUTate allows, for instance, searching for an AAL service that have to be executed, obtaining its related context situation, adding new tasks to an AAL service, and creating an AAL service.

OCean and MUTate provide access to the models by using the same vocabulary defined in the context ontology and the task model, respectively. It is important to note that, in this way, they provide high-level abstraction mechanisms that facilitate the interaction with the models without the need to stop the system. Both, OCean and MUTate are needed in order to achieve the automation and evolution of the specified AAL services.

The **context manager** monitors the pervasive sensors install in the environment to detect and process context changes. The context manager also updates the context model according to the detected context changes. The context manager uses OCean to perform this update at runtime.

The **automation engine**, named MAtE (Model-based Automation Engine), is in charge of automating the AAL services in the opportune context by interpreting the models at runtime using MUTate. To automate an AAL service, MAtE executes its system tasks by taking into account the current context,

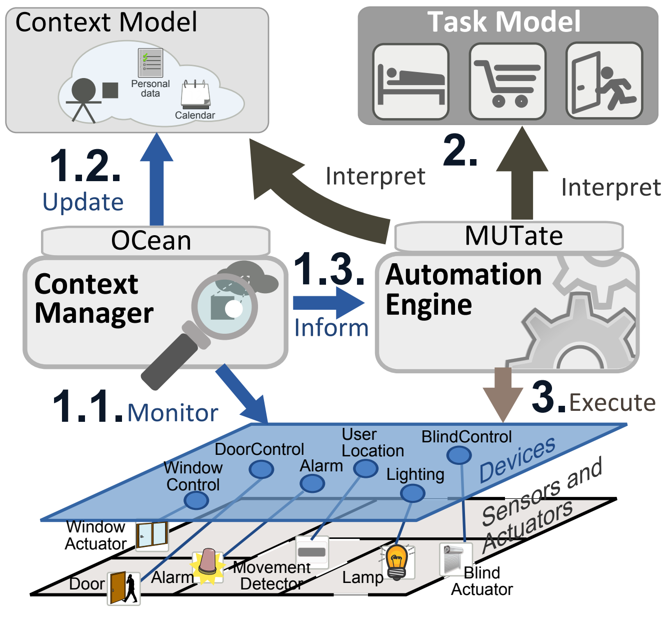


Fig. 5. Runtime infrastructure

their relationships and their refinements. Each system task is executed by MAtE using the pervasive service related to it.

A. The Software Infrastructure in Execution

The software infrastructure executes the AAL services as described in the task model by performing the following steps (see Figure 5):

1) *Detect context changes:* A context change is physically detected by a sensor. The context manager monitors all the sensors to check context changes (step 1.1 in Figure 5). For instance, the context manager monitors periodically the location of patients and whether or not a fall is detected. When a change is detected (e.g., a patient has fallen down) the context manager updates the context model using OCean (step 1.2 in Figure 5) and notifies the automation engine about the context change (step 1.3 in Figure 5).
2) *Interpret context situations:* After receiving the notification of a context change, the engine analyzes the context situations of the AAL services specified in the task model to check if any of them depend on the context change. Then, by making use of the context manager, the engine checks if any of those context situations are fulfilled. For instance, when the context manager notifies the engine that a user has fallen down, the engine gets the context situation that depends on this aspect, such as the one shown in Figure 4. If the fallen user is also alone, then the context situation of this AAL service is satisfied.
3) *Execute the AAL services:* The engine executes the AAL services whose context situation is satisfied. The engine uses the context manager to check the context conditions. To execute each AAL service, the engine executes its leaf tasks according to their refinements, their context conditions in the current context, and their temporal relationships. For instance, when the context situation of the AAL service focused on detecting falls (see Figure 4) is satisfied, the engine captures the current state, activates the emergency, and measures the health indicators by using the corresponding pervasive devices. Next, the rest of tasks are executed according to the task plan defined in the task model (see Figure 4).

If the requirements of the AAL services change over time, OCean and MUTate can be used to modify the models at runtime to perform the required evolutions. Using these mechanims, the services can be evolved by using concepts of high-level of abstraction such as user tasks, patient, or context situations. For instance, in order to make sure that an automated service executes one task instead of another, analysts just need to replace the corresponding task; in order to change the order in which tasks must be executed, analysts just need to modify temporal relationships; in order to change the situations in which services must be executed, analysts just need to change a context condition.

In addition, models are decoupled from the implementation of pervasive devices since the models just use identifiers to reference these services. This also facilitates the evolution of the assisting services since they are independent of pervasive technologies and internal implementation aspects.

B. Implementation Details

The *context manager* and the *automation engine* are implemented in Java/OSGi technology and are run in an OSGi server together with the pervasive devices.

Using OSGi, the context manager can listen to the changes produced in the services to detect context changes and can also inform the engine when a change is detected. To execute a task, the engine searches for the pervasive device associated to the task in the OSGi server by using its service registry. Then, the engine executes the corresponding device method by using the Java Reflection capabilities.

To manage the task model at runtime, MUTate uses the EMF Model Query plugin that allows a system to work with any model by querying its structure at runtime. To manage the context repository at runtime, OCean uses the OWL API 2.1.1, which provides facilities for creating, examining, and modifying an OWL model; and the Pellet reasoner 1.5.2., which allows the OWL model to be queried. More technical details can be found in [31].

VII. Validation of the Proposal

Following the guidelines provided in [32], we have developed a case-study based evaluation where the automations needed for the ACube research project were created. ACube is a project founded by the local government of the Autonomous Province of Trento in Italy. ACube aims at designing an automated user intensive system to be deployed in nursing homes as a support to medical and assistance staff.

Using the proposed methodology, we designed and developed an automated system for a specific nursing home subject of study in the ACube project. The main goal of the system was to automate AAL services that were usually performed by medical and assistance staff is to help them to make their work more efficiently in order to enhance their quality of work and the quality of life of their patients. By automating AAL services, the tasks of medical and assistance staff can be greatly reduced freeing them so that they can spend more time with their patients. In addition, the tasks that medical and assistance staff perform can be improved to be more efficient because the tasks can be previously analysed and also can be carry out even when none caregiver is present.

According to the methodology requirements (see Section III), we evaluated the following research questions:

1) Does the approach facilitate end-users participation in the AAL service design to take into account medical professional guidelines and the specific requirements from medical professionals and patients?
2) Does the approach correctly automate the specified AAL services in a context-adaptive way?
3) Does the approach allow the automated AAL services to be evolved after system deployment?

We now summarize the results of this evaluation. More details can be found in [31].

A. Evaluating End-Users' Participation

Our approach makes use of design models at runtime. It provides a task model that describes the AAL services by using concepts of a high-level of abstraction that are close to the domain and to end-users' knowledge (concepts such as task, preference, location, patient, etc.). This helps end-users to participate in the service description since it allows them to focus on the main concepts (the abstractions) without being confused by low-level details [33].

The ACube consortium had a multidisciplinary nature, involving software engineers, sociologists and analysts, and it is characterized by the presence of professionals representing end-users directly engaged in design activities. For developing the case study, we interact with some of these professionals which were responsible of analysing the requirements for the AAL services.

After describing the personas and their scenarios, we analysed them and designed the AAL services that the system should automate for supporting each one of the scenarios.

We then discussed the designed task models with the professionals in charged to include medical guidelines and validate the tasks with them. To deal with these discussions, we briefly explained the main concepts of the task model and the behaviour of two AAL services. Then, we checked the model comprehension using a short oral questionnaire that asked questions such as: how many tasks will be executed in this service?; when will this service be activated?; which is the next task that will be executed?. These questions make the users reason about the model, which is a recommended technique to evaluate the understanding of a model [34]. We found that the task model is very useful in discussing and validating the AAL services to be automated since it was very intuitive for them after explaining a couple of examples. If something was not specified the way the professionals considered suitable, we refined the model to fulfil their requirements. We repeated this process until the professionals agreed with the specification. This allowed us to describe the AAL services by taking into account the medical guidelines and requirements provided by the professionals.

Figure 4 shows the final specification of the AAL service to support patient falls. These tasks can be described as follows: the service is activated when it is detected that a patient falls and none of the caregivers or medical staff is around. When this happens, the system captures the current context state, activates the emergency state and measures the health of the patient. Then, the system alerts either the medical staff if the patient health is critical or the nearest caregivers if the patient is fine. Finally, when the emergency is under control, a report about the incidence is created and sent to the involved staff so that they can validate it.

B. Evaluating AAL Service Automation in a Context Adaptative way

To execute the described AAL services **in a context-adaptive way**, the task model describes each AAL service as a coordination of tasks that are performed in the opportune context, i.e., in a context-adaptive way. In addition, in order to be aware of the current context and to be able to automate the AAL services accordingly, the software infrastructure provides a context manager. It dynamically manages the context changes produced at runtime by using the context repository.

To validate the context adaptation, we put the system into operation to automate the described AAL services after the task models were validated with the end-users. We used a device simulator and an Equinox distribution (which is the OSGi implementation of Eclipse) running in the PC. To support the functionality needed to execute the described AAL services, we developed the required simulated pervasive devices (a total of 17 different pervasive devices). See [31] for more details about these devices.

We then evaluated the feasibility of our software infrastructure. Using the running system, we passed the JUnit tests developed to check that the specified AAL services were correctly automated as specified in the models. Since the automation of the AAL services are triggered as a response to context changes, we caused these context changes by changing the state of the sensors using the simulator. We changed the state of the sensors simulating the scenarios of the requirement elicitation phase. For instance, to enable the *Dealing with a Fall* AAL service, we simulate that a patient fell down when she was alone. This makes the context situation of the AAL service fulfil (see Figure 4).

In the same way, we simulated the rest of the scenarios of the case study and executed the prepared JUnit tests. For all of them, we checked that they were executed as specified in the models.

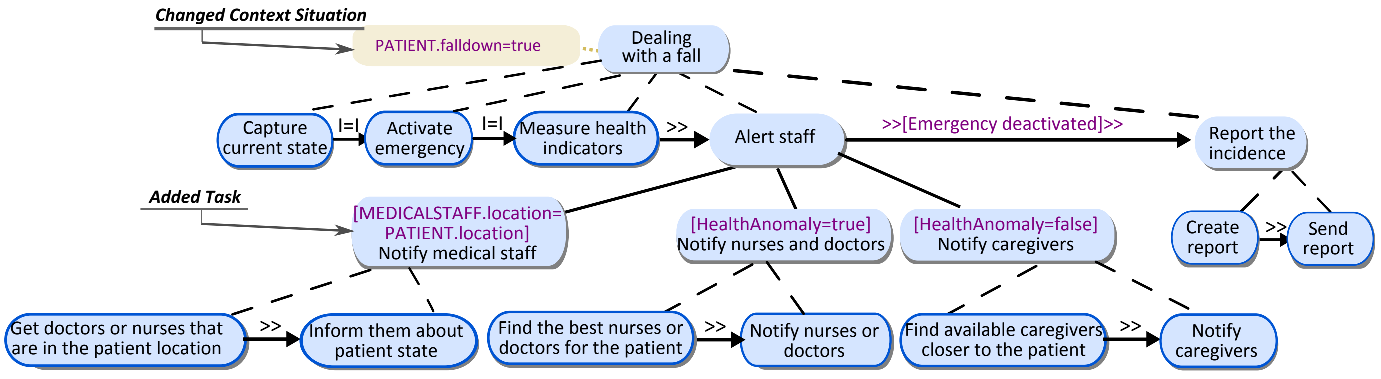

Changed Context Situation
PATIENT.falldown=true
Dealing with a fall
Capture current state
|=|
Activate emergency
|=|
Measure health indicators
>>
Alert staff
>>[Emergency deactivated]>>
Report the incidence
Added Task
[MEDICALSTAFF.location= PATIENT.location] Notify medical staff
[HealthAnomaly=true] Notify nurses and doctors
[HealthAnomaly=false] Notify caregivers
Create report
>>
Send report
Get doctors or nurses that are in the patient location
>>
Inform them about patient state
Find the best nurses or doctors for the patient
>>
Notify nurses or doctors
Find available caregivers closer to the patient
>>
Notify caregivers

Fig. 6. Example of a AAL services' evolution

Furthermore, we evaluated the system performance. Models are manipulated at runtime by OCean and MUTate. Therefore, these operations have to be efficient enough so that the system response is not drastically affected. In order to measure the system response, we quantified the temporal cost of the operation done with randomly generated large models. We used a laptop intel core i7-4600U, 2.70GHz and 8GB of RAM, with Windows 8.1 end Eclipse Modelling Kepler 32 bits. We used the context model presented in Section V and an empty task model to be randomly populated by means of an iterative process. The context model was populated with 100 new context individuals in each iteration, while the task model was populated with one new AAL service whose task structure formed a binary tree, varying the depth and width of the first level of the tree each iteration.

For each iteration, we tested all the model operations 20 times and calculated the average temporal cost of each one. As an example, the operation of OCean with the highest temporal cost was the operation to get a specific context individual, which took less than 0.2 milliseconds for 6000 individuals. The temporal cost of the MUTate model operations with the highest cost are the operations for getting, updating, and deleting a task. These costs are very similar since all of them run the same query to obtain the corresponding task. Even with a model population of 45612 tasks, these model operations provided a fast response (less than 8 milliseconds). Therefore, the results show that the response time is not drastically affected when models with a high number of instances are used.

C. Evaluating AAL Service Evolution after System Deployment

To automate the described AAL services in such a way that **their evolution after system deployment is facilitated**, the automation engine directly interprets the task model at runtime. The model is machine-processable and precise enough to be executed. Therefore, when a context change is detected by the context manager, it informs the engine. The engine then reads the AAL service information from the task model and executes the corresponding pervasive services according to context. With this strategy, the task model is the only representation of the AAL services to be automated. This allows them to be adapted by simply updating the model. As soon as it is changed to evolve the AAL services, the changes are also taken into account by the engine.

To validate this runtime evolution, we changed the task model using OCean and MUTate to perform the following types of updates: add, delete and modify tasks; modify context situations, task order, context preconditions, temporal relationships, etc.

After each update, we simulated the fulfilment of the context situations of the AAL services and applied the JUnit tests again to check that the tasks were correctly executed according to the performed evolution. For instance, Figure 6 shows an example of these evolutions. It shows how the *Dealing with a Fall* AAL service has been modified to be executed regardless if the patient is alone or not. In addition, it has been added the Notify medical staff task that is executed when there is medical staff in the same location of the patient; if so, the doctors or nurses are notified about the patient state since they can look after the patient straightaway. If there is not medical staff in the same location, then the system follows the same plan that was specified in the previous version (if any health anomaly is detected, then the appropriate doctors or nurses are notified; otherwise, the closer available caregivers are notified). For each performed evolution, we applied again the JUnit tests checking that all the AAL services were correctly executed.

VIII. Conclusions and Further Work

In this work, we have presented and evaluated a model-driven approach that achieves the automation of services for improving AAL. These AAL services are represented in high-level abstraction context-adaptive task models that are executable, i.e., they are directly executed by a software infrastructure that automates the AAL services as specified in the models. This considerably facilitates the further evolution of the AAL services by directly changing the models (i.e., at the modelling level) at runtime, which is one of the top challenges in software evolution research [35]. As soon as the models are changed to evolve the AAL services, the changes are also taken into account by the automation engine.

As further work, we plan to develop an end-user tool that allows medical professionals to create and evolve AAL services by their own. This tool will provide medical professionals with intuitive user interfaces that let them describe assisting services by using their own knowledge and concepts. Then, a model-to-model transformation will be applied in order to generate context-adaptive task models. As a previous experience in the development of this type of tools we developed an end-user tool [36] focused on the adaptation of smart home systems. This tool allows home inhabitants to create and evolve the routine tasks that they want to have automated. Our goal is to reuse all this experience to create a similar tool focused on AAL environments.

Acknowledgment

This work has funded by the Research Fund KU Leuven.

References

[1] E. Serral, P. Valderas, and V. Pelechano, "A software infrastructure for executing adaptive daily routines in smart automation environments," in *ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2013, pp. 30–35.

[2] H. Steg, H. Strese *et al.*, "Ambient assisted living–european overview report," 2005.

[3] B. Takács and D. Hanák, "A mobile system for assisted living with ambient facial interfaces," *International Journal on Computer Science and Information System*, vol. 2, pp. 33–50, 2007.

[4] A. Association. (2014) Ambient assisted living (aal) joint programme. [Online]. Available: http://www.aal-europe.eu/

[5] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman, "Creating an ambient-intelligence environment using embedded agents," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 12–20, 2004.

[6] P. Rashidi and D. J. Cook, "Keeping the intelligent environment resident in the loop," in *IE 08*, 2008, pp. 1–9.

[7] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Using context and preferences to implement self-adapting pervasive computing applications," *Software: Practice and Experience*, vol. 36, no. 11-12, pp. 1307–1330, 2006.

[8] M. García-Herranz, P. Haya, and X. Alamán, "Towards a ubiquitous end-user programming system for smart spaces," *Journal of Universal Computer Science*, vol. 16, no. 12, pp. 1633–1649, 2010.

[9] A. Cooper, R. Reimann, and D. Cronin, *About face 3: the essentials of interaction design*. John Wiley & Sons, 2012.

[10] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, "The internet of things for ambient assisted living," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. Ieee, 2010, pp. 804–809.

[11] F. G. Miskelly, "Assistive technology in elderly care," *Age and ageing*, vol. 30, no. 6, pp. 455–458, 2001.

[12] P. P. Portal. (2014) Persona project portal. [Online]. Available: http://www.aal-persona.org

[13] J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd, "The georgia tech aware home," in *CHI'08 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2008, pp. 3675–3680.

[14] U. of Illinois at Urbana-Champaign. (2014) I-living: Assisted living project. [Online]. Available: http://lion.cs.uiuc.edu/assistedliving

[15] J. Nehmer, M. Becker, A. Karshmer, and R. Lamm, "Living assistance systems: an ambient intelligence approach," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 43–50.

[16] E. Serral, P. Valderas, and V. Pelechano, "Improving the cold-start problem in user task automation by using models at runtime," in *Information Systems Development*. Springer, 2011, pp. 671–683.

[17] D. Cook and S. Das, *Smart environments: Technology, protocols and applications*. Wiley-Interscience, 2004, vol. 43.

[18] C. Pribeanu, Q. Limbourg, and J. Vanderdonckt, "Task modelling for context-sensitive user interfaces," *Interactive Systems: Design, Specification, and Verification*, pp. 49–68, 2001.

[19] R. Huang, Q. Cao, J. Zhou, D. Sun, and Q. Su, "Context-aware active task discovery for pervasive computing," in *International Conference on Computer Science and Software Engineering*, 2008, pp. 463–466.

[20] F. M. Reyes, "Issues of sensor-based information systems to support parenting in pervasive settings: A case study," *Emerging Pervasive and Ubiquitous Aspects of Information Systems: Cross-Disciplinary Advancements*, p. 261, 2011.

[21] J. Muñoz, E. Serral, C. Cetina, and V. Pelechano, "Applying a model-driven method to the development of a pervasive meeting room," in *ERCIM News*, April 2006, pp. 44–45.

[22] E. Serral, P. Valderas, and V. Pelechano, "Towards the model driven development of context-aware pervasive systems," *Special Issue on Context Modelling, Reasoning and Management of the Pervasive and Mobile Computing (PMC) Journal*, 2010.

[23] E. Serral, P. Valderas, and V.Pelechano, "Context-adaptive coordination of pervasive services by interpreting models during runtime," *The Computer Journal*, vol. 56, no. 1, pp. 87–114, 2013.

[24] (2014) Osgi. http://www.osgi.org/.

[25] A. Sutcliffe, N. Maiden, S. Minocha, and D. Manuel, "Supporting scenario-based requirements engineering," *IEEE Trans. on Soft. Eng.*, pp. 1072–1088, 1998.

[26] C. Rolland and C. Salinesi, "Supporting Requirements Elicitation through Goal/Scenario Coupling," in *Conceptual Modeling: Foundations and Applications*. Springer, 2009, p. 416.

[27] A. Cooper, R. Reimann, and D. Cronin, *About face 3: the essentials of interaction design*. Wiley India Pvt. Ltd., 2007.

[28] E. Serral, L. Sabatucci, C. Leonardi, P. Valderas, A. Susi, M. Zancanaro, and V. Pelechano, "Incorporating users into ami system design: From requirements toward automation," in *Information Systems Development*, R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, and M. Lang, Eds. Springer New York, 2013, pp. 499–511.

[29] Smith, Welty, and McGuinness, "Owl web ontology language guide," 2004.

[30] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, M. A. Musen *et al.*, "Protege-2000: an open-source ontology-development and knowledge-acquisition environment," in *AMIA Annu Symp Proc*, vol. 953, 2003, p. 953.

[31] E. Serral, "Automating routine tasks in smart environments. a context-aware model-driven approach," Ph.D. dissertation, Technical University of Valencia, DSIC, 2011.

[32] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[33] F. Paternò, "From model-based to natural development," *HCI International*, pp. 592–596, 2003.

[34] A. Gemino and Y. Wand, "Evaluating modeling techniques based on models of learning," *Communications of the ACM*, vol. 46, no. 10, October 2003, modeling comprehensibility.

[35] T. Mens, "The ercim working group on software evolution: the past and the future," in *IWPSE-Evol workshops*. ACM, 2009, pp. 1–4.

[36] E. Serral, F. Pérez, P. Valderas, and V. Pelechano, "An end-user tool for adapting home automation to user behaviour at runtime," *UCAmI'10*, pp. 201–210, 2010.