Service Composition and Advanced User Interfaces in the Home of Tomorrow: The SM4All Approach

Tiziana Catarci, Claudio Di Ciccio, Vincenzo Forte, Ettore Iacomussi, Massimo Mecella, Giuseppe Santucci, and Giuseppe Tino

Sapienza – Università di Roma,
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Via Ariosto 25, Roma, Italy
{catarci,cdc,mecella,santucci}@dis.uniroma1.it,
name.surname@gmail.com

Abstract. Houses of tomorrow will be equipped with many sensors, actuators and devices, which collectively will expose services. Such services, composed in an automatic way, and invokable through adaptive user interfaces, can support human inhabitants in their daily activities. In this paper we present the approach and some results of the SM4All EU project (http://www.sm4all-project.eu/), which is investigating automatic services composition and advanced user interfaces applied to domotics.

Keywords: SM4All, smart devices, advanced user interfaces, service composition, domotics.

1 Introduction

Embedded systems, i.e., specialized computers used in larger systems to control equipment are nowadays pervasive in immersive realities, i.e., scenarios in which, invisibly, they need to continuously interact with human users, in order to provide continuous sensed information and to react to service requests from the users themselves. Examples are digital libraries and eTourism, automotive, next generation buildings and domotics. Sensors/devices/appliances/actuators offering services are no more static, as in classical networks, (e.g., for environmental monitoring and management or surveillance), but they form an overall distributed system that needs to continuously adapt. That can be done by adding, removing and composing on-the-fly basic elements that are the offered services.

This paper intends to outline some insights stemming from the European-funded project SM4All (Smart hoMes for All - http://www.sm4all-project.eu/), started on September 1st, 2008 and finishing on August 31st, 2011. SM4All aims at studying and developing an innovative platform for software smart embedded services in immersive environments, based on a service-oriented approach and composition techniques. This is applied to the challenging scenario

S. Gabrielli, D. Elias, and K. Kahol (Eds.): AMBI-SYS 2011, LNICST 70, pp. 12–19, 2011.

of private home and building in presence of users with different abilities and needs (e.g., young, elderly or disabled people).

In order to introduce the novel idea of services underlying SM4All, the reader should consider the following scenario: a person is at home and decides to take a bath. He/she would like to simply express this to the house and have the available services collaborate in order to move the house itself to a new state which represents the desired one. The temperature in the bathroom is raised through the heating service, the wardrobe in the sleeping room is opened in order to offer the bathrobe, the bath is filled in with 37 ^{o}C water, etc. If we suppose the person is a disabled one, some services cannot be directly automated, e.g., the one of helping the person to move into the bath. In this case, a service still exists, but it is offered by a human, e.g., the nurse, which is doing her job in another room, and that at the right moment is notified – through her PDA or any other device – to go into the bath and help the patient. Maybe this service is offered also by the son of the patient (or any other person), living in a nearby house, which is notified at the right moment, and if the nurse is not present at home, to help the patient. The scenario shows the idea of a system of services, some offered in a completely automated way through sensors/appliances/actuators, other realized through the collaboration of other persons, which moves continuously from a desired state to a new one, in order to satisfy user goals. Clearly, as in all complex systems, there are trade-offs to be considered (the goal of the person willing a relaxing bath is in contrast with the availability of the nurse/son offering the "help" service).

The rest of the paper is organized as follows: Section 2 does very briefly provide a background on the current state of the art for home automation systems, Section 3 gives the reader an overview of the SM4All system architecture. For sake of space, the remainder of the paper then focuses only on two components, namely the User Interface (Section 4) and the Composition Layer (Section 5), which are among the most innovative ones produced by the project. Finally, the Section 6 draws some conclusions.

2 Relevant Work

Presently, we are assisting at a blooming of research projects on the use of smart services at home and domotics, in particular for assisting people with physical or mental disabilities.

For instance, at Georgia Tech a domotic home has been built for the elder adult with the goals of compensating physical decline, memory loss and supporting communication with relatives [9]. This work also considers issues of acceptability of domotics identifying key issues for the adoption of the technology by the end user. Acceptability, dangers and opportunities are also surveyed in [13]. Having a reliable system is a primary concern for all users.

At Carnegie Mellon people's behavior is studied by automatic analysis of video images [4]. This is fundamental in detecting anomalies and pathologies in a nursing home where many patients live. Pervading the environment with active landmarks, called Cyber Crumbs, aims at guiding the blind by equipping

him/her with a smart badge [14]. A number of projects to give virtual companion's to people, to monitor people's health and behavioral patterns, to help Alzheimer patients are presented in [6]. The Gator Tech Smart House [5] is a programmable approach to smart homes targeting the elder citizen. The idea is to have a service layer based on OSGi [15] in order to enable service discovery and composition.

Finally, in [10], the current adoption of service technologies for smart energy systems, including domotic ones, is discussed.

3 The SM4All Architecture

Devices and sensors available in the house constitute the basis of the SM4All system. There is an ever increasing variety of devices, for example for controlling parts of the home (doors, lights), or media devices, etc. Sensors are devices for measuring physical quantities, ranging from simple thermometers to self-calibrating satellite-carried radiometers. Sensors and devices have an inherent connection, e.g., a device for opening the window blinds can change the luminosity value sensed by a sensor. Both sensors and devices make their functionalities available according to the service oriented paradigm. In particular, services (offered by sensors and devices) are offered as SOAP-based services, both in the UPnP technology and in the WSDL-based one.

A *Pervasive Controller* is in charge, when a new sensor/device joins the system, to dynamically load and deploy the appropriate service wrapping it, and to register all the relevant information into the *Service Repository*. Each service, in order to be dynamically configurable and composable, exposes rich service descriptions, comprising *(i)* interface specifications, *(ii)* specifications of the externally visible behaviors, *(iii)* offered QoS. As previously introduced, human actors in the environment are also abstracted as services, and actually "wrapped" with a rich description (e.g., a nurse offering medical services). This allows including them in a service composition and having them collaborate with devices and sensors to reach a certain goal. Such rich service descriptions are stored in the Service Repository, in order to be queried and retrieved by the other components of the architecture. During their operation, services continuously change their status, both in terms of values of sensed/actuating variables (e.g., a service wrapping a temperature sensor report the current sensed temperature, a service wrapping windows blinds report whether the blinds are open, closed, half-way, etc.) and in terms of conversational state of the service itself. All these status information are available, through a publish&subscribe mechanism, in the *Context Manager*.

On the basis of the rich service descriptions, a *Composition Engine* is in charge of providing complex services by suitably composing available ones. The engine can work in two different ways: *(i)* off-line and *(ii)* on-line. In the off-line mode, at design/deployment time of the house, a desiderata (i.e., not really existing) target service is defined, and the composition engine (through its synthesis subcomponent) synthesizes a suitable orchestration of the available services realizing the target one. Such an orchestration specification is used at execution-time

(i.e., when the user chooses to invoke the composite/desiderata service) by the orchestration subcomponent in order to coordinate the available services (i.e., to interact with the user on one hand and to schedule service invocations on the other hand). In this mode, the orchestration specification is synthesized off-line (i.e., not during user requests) and executed on-line as if it were a real service of the home. The off-line mode is technically based on the so-called Roman approach to automatic service composition [1]. Conversely in the on-line mode, the user, during its interaction with the home, may decide not to invoke a specific service (either available/real or composite one), but to ask the home to realize a goal; in such a case, the composition engine, on the basis of specific planning techniques [8], synthesizes and executes available service invocations in order to reach such a goal.

The user is able to interact with the home through different kinds of *user interfaces*, either centralized (e.g., in a home control station accessible through a touchscreen in the living room) or distributed and embedded in specific interface devices. Specifically, Brain Computer Interfaces (BCIs) allow also people with disabilities to interact with the system.

4 The User Interface

The home can be controlled from the user through different kinds of interfaces (BCIs, remote controls, touch screens, keyboards, voice recognition, etc.). The *AAI* (Abstract Adaptive Interface) represents the core of the SM4All user layer. It retrieves status information from the Context Manager and the service descriptions from the Service Repository and organizes everything in order to correctly show available actions to the user, depending on the interaction mode she is currently making use of (i.e., visual, aural, BCI, etc.). The AAI is intended indeed to be put as an abstraction layer among the multiple user interface devices and the underlying composition layer.

Its main novelty is represented by the ability to manage many different user interface models with a unique adaptable algorithm, able to change itself on the basis of the interaction device characteristics (speech/aural, visual/touch, handheld, brain-controlled...) and on the basis of the user preferences, automatically gathered, analyzed and synthesized on top of the previous interactions with the system.

Through a message screen the user can see notifications coming from the system. The room actions' screen shows the list of actions that can be invoked, gathered up by groups which are built according to the rooms where the services offering the actions are actually located. The number of available services in the home can be very high, and a service can offer many actions; on the other hand, the icons that can be shown on a screen is limited. A pagination of the information, though useful and indeed exploited in many prototypes, is not sufficient to provide an effective interaction, since it would introduce a huge effort for the user to find the desired element among the big amount of items, navigating back and forth. Hence, in SM4All, the AAI integrates a mechanism for grouping and smartly ordering the icons in order to improve the ease of interaction. An

icon may represent either a service or an action. Sometimes, only a few actions, among the ones offered by a given service, are available, e.g., a "bedroom light" service offers a "turn off" and a "turn on" action, but only the first (or the second, conversely) is available when the lamp is switched on (off). In such a case, there is no need to show the service icon, as the available action is enough.

When the user can fire more than one action, related to a single service, a clustering is needed. It is realized by initially showing the service icon; once activated, all of the other items are hidden and only the available related actions are displayed.

Another way to reduce the number of displayed icons is to divide services themselves into groups represented by a given type (e.g., "Multimedia" for TVs, MP3 players, etc.). The idea is almost the same: at first, the menu shows only a type which many services belong to, and then, after the type is selected, all of the other items are hidden and the only related services are displayed (see Figure 1).

Fig. 1. An example of the user interface grouping services by the type they belong to

Beyond grouping, the AAI exploits the possibility to order the items according to their importance, with respect to the preferences of each user. This way, the actions which are known to be more relevant for the user will be displayed on the first screen, in order to appear at a first glimpse, while the others are going to be shown next. Two algorithms are offered: a *static* one and a *dynamic* one. The user can select which one she prefers through an administration menu. The static algorithm makes use of explicitly defined user settings to identify her preferences. Each preference is constituted by *(i)* a set of conditions, representing the state of the environment which enables the action, *(ii)* a time frame in which the preference has to be considered (Always, Morning, Afternoon, Evening, Night), and *(iii)* a usage expectation degree (certain, highly probable, very probable, probable). The dynamic algorithm orders the actions according to the probability that each one is going to be executed, on the basis of the current environment status and previous invocations: the higher the probability, the higher the priority of the associated icon in the list (*partial order*). The home environment status consists of several parametric values related to the execution (e.g., the 24-hour format time of invocation). Each parameter is associated to a

relevance (*weight*), manually tunable by administrators. At every call, the parametric value is computed and its *incidence* (*score*) re-calculated. Indeed, it is taken from a run-time updated graph, i.e., a normalized sum of Gaussian curves: at each execution, a new Gaussian centered in the parameter value which is associated to the call (e.g., the time of invocation) is added to the previous graph. In order to tune the evolution of the curve, norm and variance of the Gaussians are both customizable. If the global peak overcomes the maximum admissible value (100%) a normalization is automatically performed (see Figure 2). The probability is thus the *sum* of the *weighted scores* ($\sum_i relevance_i \times incidence_i$) of all the parameters, related to the current home environment status.

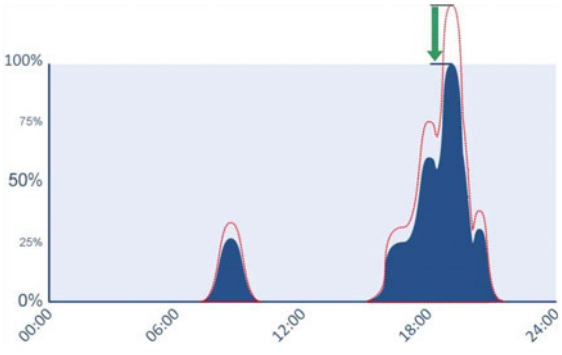

Fig. 2. An example of automatic normalization of the parametric score

5 Composition

The Composition Engine is the component in charge of creating orchestrations which control the whole set of available services spread around the house, in order to satisfy the users' desiderata. It is divided into two different functional components, On-line Synthesis Engine and Off-line Synthesis Engine. As the name suggests, the former performs such a computation at run-time, i.e., as the user asks for a new plan it must return an orchestration to be invoked immediately after. This orchestration is a program that the *Orchestration Engine* must follow step by step in order to achieve the goal. The representation of services is based upon the listing of their pre- and post-conditions, i.e., logic formulae on top of environmental variables such as a room temperature, or light level, which, respectively, *(i)* must be verified for an action to take place and *(ii)* must hold after the action is completed. Goals are logic formulae on top of the same environmental variables, which the user expects to become true due to the enactment of the synthesized plan. The On-line Synthesis Engine reasoning core is a planner that, as described in [7], solves the underlying planning problem through the reduction of it into a CSP (Constraint Satisfaction Problem – see [3]).

In the Off-line Synthesis Engine, services and goals are both descibed by Finite State Automata (FSAs), as in [1]. Goals are target FSAs which the Composition

Engine must realize by simulation, on top of the community of available services. Pre- and post-conditions are expressed as constraints over the FSA transitions (see [11] and [2]), on top of environmental variables. Once the orchestration is computed, the target itself is stored among the community: it can be invoked at any time in the future, like any other service. Such synthesis is fulfilled within a background process by the Composition Engine. The returned orchestration is slightly different from the On-line Synthesis Engine output. Indeed, it is a relation that, given the current target state and the next action to be fired, indicates which services in the community can be invoked in order to enact it, according to any of the possible (i.e., coherent with the realizability of the goal) community and environmental variables states. The Off-line Synthesis Engine solution approach is based on reducing the problem to the synthesis of Linear-time Temporal Logic (LTL) formulae (see [12]) by Model Checking over Game Structures.

The Off-line Synthesis Engine service automaton description language, as well as the orchestration structure definition language, which are used to interact with the component, are XML-based, as a common attitude in service-oriented distributed systems. Their schemata are published under public addresses, so to become a clear interface language not only for the consortium developers but also for external vendors who might want to sell products which are compliant to a SM4All standard[1]

As far as the environmental variables are concerned, their types are standardized as well, under a unique data schema. Following the rationale used for services and orchestrations descriptions, it is specified by the definition of hierarchical types in an XML Schema format. Essentially, the SM4All data schema imposes a restriction over native XML Schema simple types to limit them to totally ordered countably finite data sets on top of which the developer can define enumerations or intervals, so to make the reasoning over such data feasible[2].

The Repository (see Section 3) is in charge of storing all of XML descriptors. The conversion from such XML formats to internal reasoners input/output formats are up to façade modules which are not exposed to the outside components in the SM4All architecture.

6 Concluding Remarks

Throughout this paper, we presented the pervasive intelligent home system SM4All, and we focused on the synthesis techniques adopted by its Composition Engine and on the self-adapting ones of the User Interface: they are the components involved the most in the challenge of hiding the heterogeneity of used hardware devices to the other software modules, which is a very important requirement in the field of domotics, where a lack of standardization holds still.

[1] The reader can access service and orchestration XML Schemata and related documentation at `http://www.dis.uniroma1.it/~cdc/sm4all/proposals/servicemodel/`

[2] The reader can find the Schemata and documentation at `http://www.dis.uniroma1.it/~cdc/sm4all/proposals/datamodel/`

Indeed, this paper also introduces the proposed XML format for the definition of services offered by home devices.

Currently, we are developing a running prototype interfaced with real Konnex, UPnP and Bluetooth devices actually installed in a house set up on purpose in Rome, hosted by Fondazione Santa Lucia. A showcase will be demonstrated in May and October 2011; this will make it possible for us to gather and analyze experimental results in the usage of the new system running.

References

1. Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M., Patrizi, F.: Automatic Service Composition and Synthesis: the Roman Model. IEEE Data Eng. Bull. 31(3), 18–22 (2008)
2. De Masellis, R., Di Ciccio, C., Mecella, M., Patrizi, F.: Smart Home Planning Programs. In: Proc. of 2010 International Conference on Service Systems and Service Management (ICSSSM 2010). Japan Advanced Institute of Science and Technology, Japan (2010)
3. Do, M., Kambhampati, S.: Solving Planning-Graph by Compiling it into CSP. In: Proc. AIPS 2000, pp. 82–91 (2000)
4. Hauptmann, A., Gao, J., Yan, R., Qi, Y., Yang, J., Wactlar, H.: Automatic analysis of nursing home observations. IEEE Pervasive Computing 3(2), 15–21 (2004)
5. Helal, S., Mann, W.C., El-Zabadani, H., King, J., Kaddoura, Y., Jansen, E.: The gator tech smart house: A programmable pervasive space. IEEE Computer 38(3), 50–60 (2005)
6. Joseph, A.: Successful aging. IEEE Pervasive Computing 3(2), 36–41 (2004)
7. Kaldeli, E.: Using CSP for Adaptable Web Service Composition. Tech. Rep. 2009-7-01, University of Groningen (2009), `www.cs.rug.nl/~eirini/tech_rep_09-7-01.pdf`
8. Kaldeli, E., Lazovik, A., Aiello, M.: Extended Goals for Composing Services. In: Proc. 19th International Conference on Automated Planning and Scheduling, ICAPS 2009 (2009)
9. Mynatt, E., Melenhorst, A., Fisk, A., Rogers, W.: Understanding user needs and attitudes. IEEE Pervasive Computing 3(2), 36–41 (2004)
10. Paradiso, J., Dutta, P., Gellersen, H., Schooler, E.: Smart energy systems. special issue. IEEE Pervasive Computing 10 (2011)
11. Patrizi, F.: Simulation-based Techniques for Automated Service Composition. Ph.D. thesis, Department of Systems and Computer Science, SAPIENZA - Universitá di Roma, Rome, Italy (2009)
12. Pnueli, A., Rosner, R.: On the Synthesis of a Reactive Module. In: Proc. POPL, pp. 179–190 (1989)
13. Roberts, J.: Pervasive health management and health management utilizing pervasive technologies: Synergy and issues. The Journal of Universal Computer Science 12(1), 6–14 (2006)
14. Ross, D.: Cyber crumbs for successful aging with vision loss. IEEE Pervasive Computing 3(2), 30–35 (2004)
15. Tuecke, S., Foster, I., Frey, J., Graham, S., Kesselman, C., Maquire, T., Sandholm, T., Snelling, D., Vanderbilt, P.: Open service grid infrastructure (2003)