

A user-oriented and context-aware service orchestration framework for dynamic home automation systems

Thomas Coopman, Wouter Theetaert, Davy Preuveneers and Yolande Berbers

Abstract The home environment has become increasingly sophisticated. Sensors and actuators allow a house to be *programmed* so that it can intelligently adapt to user needs. Unfortunately it has not become any easier for the user to manage all these devices. Our ontology driven framework describes a user-oriented automation system to help with this process by not only orchestrating home automation devices, but also by integrating new equipment and automating them as much as possible. By combining devices with a common functionality in a given context, relevant devices are pulled together in an intuitive user interface in which one can express preferences at a high level of abstraction. The big advantage is that our framework allows for the home environment to be adjusted without the user having to know about the technical details of devices in the home automation system.

Key words: home automation, context-awareness, service composition

1 Introduction

Computers are becoming smaller and more powerful. The image of the computer only used as a desktop is becoming obsolete [12]. Today computers are everywhere, even in our home environment: lighting controlled by light sensors, garage doors that open and close automatically, curtains that close automatically in the evening, time triggered coffee makers. We see that technological advances have equipped household appliances with more and more functionality and flexibility. The other

Thomas Coopman and Wouter Theetaert
Department of Computer Science, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, e-mail:
firstname.lastname@student.kuleuven.be

Davy Preuveneers and Yolande Berbers
Department of Computer Science, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, e-mail:
firstname.lastname@cs.kuleuven.be

side of the medal is that the manuals on how to use home automation systems are becoming bigger and bigger. It is a daunting task to understand all the possibilities and capabilities. Furthermore, many sensors or actuators will need to work together, and configuring such installations will not be easy. Collecting and managing context information [6] from the vicinity is becoming more and more practical every day. If context information were also to be used during the interaction between a person and the home automation system, then the human-machine interaction would become a more user friendly process [10].

In this paper we describe an ontology-driven system that addresses the above concerns. Our dynamic, context-sensitive composition framework leverages service orchestration [8] to improve the interoperability between various devices in a home automation system and provides an interface that will make life easier for the end user. Sensors and actuators are enhanced with a semantic description of their capabilities. The framework uses context information to offer management and automation facilities to the user at the highest level of abstraction possible. In this way the user has an intuitive interface to adjust the system to his own preferences.

After discussing related work in section 2, we continue in section 3 with a motivating scenario illustrating how our context-aware service orchestration framework for home automation works. In section 4 we discuss the semantic representation and mapping in our framework. The architecture of the framework is described in section 5. We end with some conclusions and topics for future work in section 6.

2 Related work

Before we dive into the design of our framework, we first discuss existing systems that use context-sensitivity and/or service composition as key building blocks. We highlight potential concerns and discuss elements reused in our own framework.

In [2], Chen and Kotz present an overview of applications that use context-sensitivity to their advantage. There are applications that forward phone calls to a device the closest to the user, tourist guides that offer a word of explanation about artefacts in the vicinity [4], and several others. Many applications, however, do not make sufficient use of the offered possibilities. They confine themselves mainly to the use of information about location and time. First, simple applications respond to context sensors that capture the presence of people in the environment, temperature, light intensity, sound, etc. Our system needs to infer complex situations from the low level context (e.g. *a person is in a meeting room, there are other people present, there is noise* could mean there is a meeting in progress). Second, context-sensitive systems often use a traditional application model, where the context dependencies are modeled with a lot of *if-then-else* rules that specify how the program must respond to a particular input.

The above approach has obvious disadvantages: when new context sensors or actuators are added in the home environment, old rules may have to be rewritten as some of the rules may now conflict with one another or have become ambiguous.

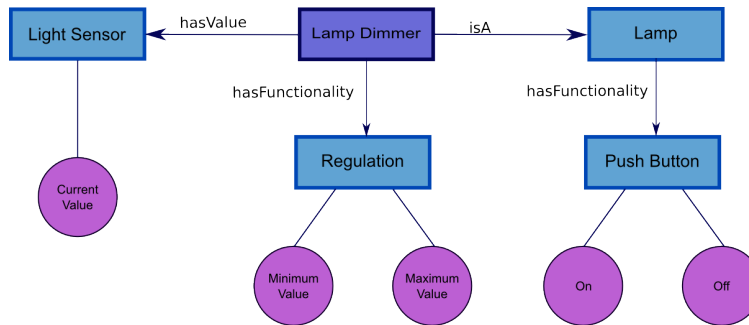


Fig. 1 A graphical representation of an ontology for a lamp dimmer.

Also, the approach is not flexible for a home automation system as the end user is not a developer and he should not be one. The user needs to set his goals in a simple way, without technical knowledge of the various services he uses. As such, user intentions should be declarative: the user should state what should be achieved (e.g. moderate lightning), but not how this should happen (e.g. change the light intensities of the individual light bulbs). With each component being semantically enhanced, it is also much easier for a new device to be automatically integrated into the system. The home automation system presented here uses many context elements with ontology- and rule-based inference to dynamically adjust the composition of sensors and actuators, by automatic translation of highlevel goals into low-level device orchestrations and vice versa. In the literature different ontologies to describe context and services have been proposed. For representing context, we have used and extended SOUPA [3]. Two ontologies that are useful to model services within the home automation environment are Domo-ML [11] and DogOnt [1]. These ontologies are quite complete, but lack some useful information to enable dynamic composition. In traditional service composition systems, such as in web services [7, 9], the interconnection of two blocks is often based on a simple input / output agreements. To make sure that only compatible devices are linked, their descriptions need to match semantically. The DogOnt ontology has been extended for this purpose. Fig. 1 shows how a lamp dimmer in DogOnt can look like.

3 Motivating scenario and requirements

In this section we describe how our framework operates by means of a motivating scenario. Suppose Alice and Bob recently bought a home automation system that uses our framework. Upon startup, the system will be searching for devices (sensors, actuators and controllers) present in the house. When Bob arrives home from work, he likes to watch the news headlines on television. He takes the remote control with on display and connects to the automation system. Since this is a new

Table 1 Services with a similar functionality

Service	Category	Service	Category
Spot lights	Light	Shutters	Security
Reading lamp	Light	Door	Security
Light sensor	Light	Radio	Music
Shutters	Light	CD player	Music

system, there are no compositions set. Bob chooses to create a new composition, and will see a list of potential building blocks for the composition. Bob chooses the categories *light*, *video* and *video* he wants to configure when he wants to watch the news (see Fig. 2). Bob wants moderate light, neutral sound and the news on his favorite channel. The system will convert the high-level settings of the user to specific low-level instructions for the various devices. Moreover the system ensures that the devices are linked together. If Bob would move, the composition follows him along. So the video of the news is moved to the nearest screen, and the sound to the closest sound system in the neighborhood. Alice comes home after having bought a new reading lamp for the living room. The system detects the light and – based on the lamp’s semantic description – adds it directly to the category *light*. When Bob wants to watch the news afterwards, he can recall the composition that he just made. The light setting *moderate* will be re-activated and the settings of the lamps will be adjusted for the fact that a new lamp is added.

All resources should be classified according to their semantic description into blocks with similar functionality (see Table 1). Thus, all lighting devices should be classified in the right category *Light*. Note that certain services can belong to different categories: e.g. the shutters can serve a purpose when controlling the lightning, but also for the security of the house. The list of building blocks shown to the user should be context-dependent: the block *audio* should only be displayed when Bob is in a room in which devices are present that can produce sound. Each of the selected blocks can be configured separately at a high level. The system described above should include some special features. It should make use of information from the environment: the system must show only blocks that have some utility in the current context. The user must also have no knowledge of the various individual devices in the system. He should be able to enter his preferences at a high level and the system should make the desired composition. There may of course be some predefined compositions in the system, but the user should have sufficient possibilities to customize everything to his desires.

4 Mapping of user settings to low-level instructions

It is now clear that our framework needs to automate a lot of things without user intervention. The intention is that a user can give high level goals while the framework converts them into low-level instructions. To meet these requirements we need

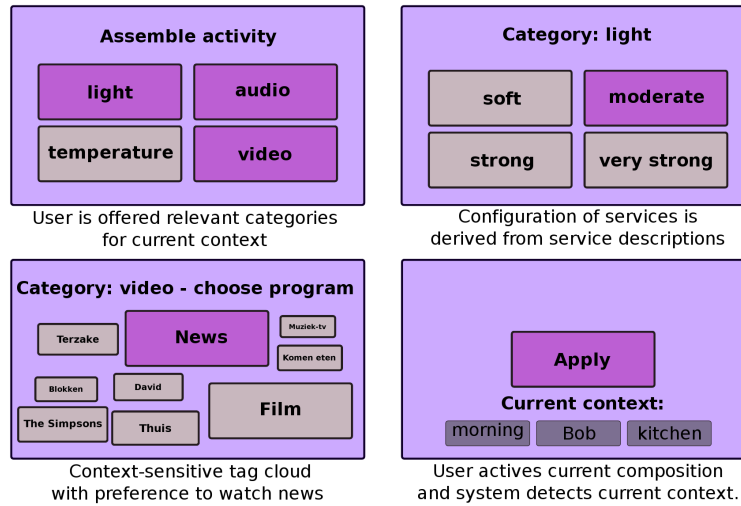


Fig. 2 Illustrations of the interface of the framework to the end user.

to represent context and services to characterize the various possibilities of the home automation system. Such representation must meet two conditions:

- First, the representation must be sufficiently expressive and extensible: different services must be linked and this in a 'smart' way. Furthermore, services may come and go as old devices are being replaced. This is more than merely linking two blocks where the input of one block corresponding to the output of another block. The link must also be semantically correct. It makes no sense to control the lightning with a temperature sensor.
- The second condition is that the representation must allow for information reasoning to model common sense knowledge about home automation systems. Such a system that works with sensors must assume that people may sometimes provide incomplete or incorrect information. Conflicts that may arise have to be resolved by the context model.

Due to these requirements, we have extended our context ontology [5] and also use an ontology based model (see Fig. 1) to model home automation knowledge and information. Our framework relies heavily on subsumption relationships between classes and properties in ontologies to detect incompatibilities. Common sense knowledge is expressed with semantic rules that are processed by the Hermit Web Ontology Language (OWL) reasoner¹. For example, during discovery we may be looking for a service that will need to interact with another given service. Therefore, the interfaces of the cooperating services should match. This is illustrated in the service composition in Fig. 3. Service 2 delivers its output of type

¹ <http://hermit-reasoner.com/>

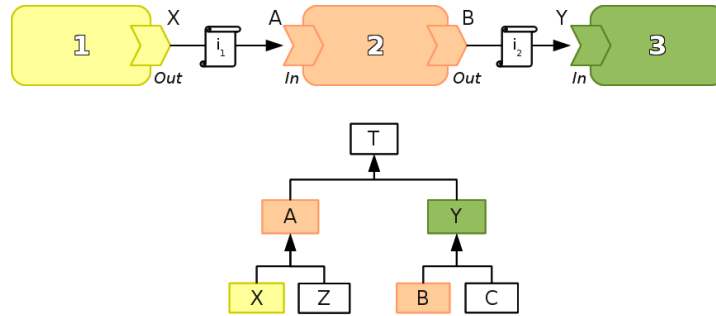


Fig. 3 Semantically matching input/output interfaces and categories in a service orchestration

B to service 3 and requires input of type A from service 1. Any instance i_1 (here of type X) whose type is subsumed by type A will match. Likewise should the type of instance i_2 (here of type B) be subsumed by type Y . We use subsumption in a similar way to find devices with specific capabilities.

An important aspect of the implementation is the conversion between high- and low-level contextual values. First, the low-level configuration possibilities of the devices should be presented to the end-user in a easy way (see Fig. 2). For that reason, the concept 'mapping' was introduced. In the ontology, each block is connected with a specific mapping. This mapping determines how the end-user can set the values for that particular block. Second, the high-level configurations of the end-user have to be enforced at the level of the devices. When an end-user asks for 'moderate light', this should be translated into practical values of the various lighting devices in the room. While doing this, the actual value of a certain property should be taken into account: when 'moderate light' is asked and this matches with some lighting value, the actual light value in the room should be measured before deciding whether the lights or the shutters should be activated.

5 Architectural overview of the home automation framework

The architecture of the system consists of several components as can be seen in Fig. 4. On the figure are two major parts: the internal system on the left, and external entities (services, context sensors, users via user interfaces) on the right. All communication passes through the *Communications Manager*. It communicates with all sensors and actuators in their own protocol and ensures that the other devices should not care about different protocols being in use. There are three types of information flows that enter the system: a device that sends information about its services (e.g. coffee), context sensors that provide information about the current sit-

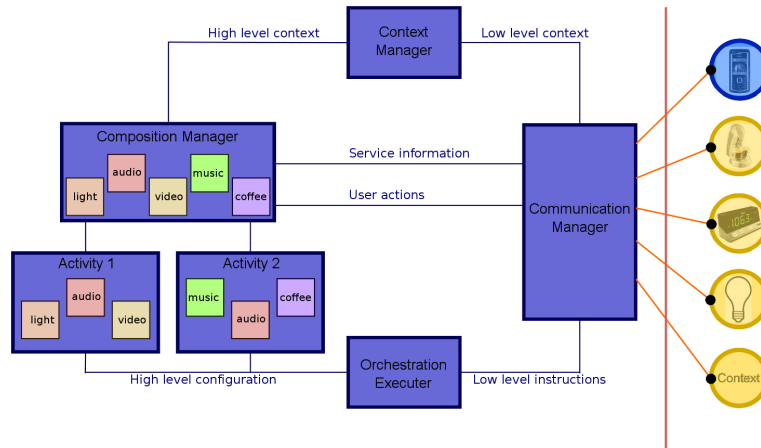


Fig. 4 Architectural overview of the home automation system

uation (e.g. a light sensor), or users that provide information through a user interface (e.g. an interactive display in the house) to change or activate some settings.

The information from the sensors (the low-level context) is sent to the *Context Manager* which reasons and infers high-level context, which is then forwarded to the *Composition Manager*. For example, a Bluetooth sensor reports the presence of a smartphone. The *Context Manager* then finds where the sensor is located (the living room) and who owns the smartphone (Bob). This way high-level context information is derived (Bob is in the living room).

Besides high-level context information, the *Composition Manager* also obtains information about the various services in the system, and this in the form of ontologies. As mentioned earlier, the services are grouped into blocks with similar functionality (e.g. light, audio, video, etc.). The user can use these blocks to assemble a new composition. The *Composition Manager* helps by just showing the blocks that are currently relevant (those whose devices are currently available).

Finally, the composition needs to be enforced for the various services. The *Orchestration Executor* is responsible for converting the information of the various blocks into the service specific implementations. This conversion is only possible if the ontologies of the services contain enough information about their control and data flow.

6 Conclusions and future work

This paper addresses the concern of managing complex dynamic home automation systems. More and more devices enter the home environment, and it is becoming increasingly more difficult to install and use them. Moreover is not evident for sev-

eral sensors and actuators to work together out of the box. Furthermore, not every connection makes sense (you don't want to control the heating with a light sensor). In addition, we want to enable the user to manage the composition rather than the manufacturer of the system. Therefore, it is obvious that managing the home automation system should be done in an intuitive way. We described a motivating scenario and examined to what extent existing solutions could help. Many existing context-sensitive systems do not make sufficient use of context and are implemented in a very static way. In almost all service composition tools, the orchestration is carried out by a developer based on simple input / output match. This is inconsistent with our expectations that the composition should be configured declaratively by the end user rather than in an imperative manner by the developer. Finally, we discussed the architecture of our context-aware service orchestration framework that supports such an approach for dynamic home automation systems.

As future work, we envision extensions to the system in the form of some predefined composition templates that can easily be changed by the user. These could be useful for modeling recurring circumstances, for example if nobody is in the house (the lights are off, all devices are switched off, the alarm is set). Furthermore, we are also thinking of extending our framework with a learning component so that the compositions are not only explicitly defined by the end users, but can be implicitly defined according to learned patterns in user behavior.

References

1. D. Bonino and F. Corno. Dogont - ontology modeling for intelligent domotic environments. *Lecture Notes on Computer Science*, pages 790–803, 2008.
2. G. Chen and D. Kotz. *A Survey of Context-Aware Mobile Computing Research*. Department of Computer Science, Dartmouth College, 2000.
3. H. Chen, T. Finin, and A. Joshi. The soupa ontology for pervasive computing. *Ontologies for Agents: Theory and Experiences*, pages 233–258, 2005.
4. K. Cheverst, K. Mitchell, and N. Davies. Design of an object model for a context sensitive tourist guide. *Computers & Graphics*, 23:883–891, 1999.
5. D. Preuveneers, et al. Towards an extensible context ontology for Ambient Intelligence. In *2nd European Symposium on Ambient Intelligence*, volume 3295 of *LNCIS*, pages 148 – 159, 2004.
6. A.K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.
7. OASIS. Web services business process execution language version 2.0, 2007. <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>.
8. M. P. Papazoglou and D. Georgakopoulos. Service oriented computing. *Commun. ACM*, 46(10):24–28, October 2003.
9. J. Rao and X. Su. A survey of automated web service composition methods. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54, 2004.
10. A. Schmidt. Implicit Human Computer Interaction Through Context. In *Personal and Ubiquitous Computing, Vol 4 (2/3)*, 2000.
11. L. Sommaruga, A. Perri, and F. Furfari. Domoml-env: an ontology for human home interaction. *SWAP 2005: Proc. of the 2nd Italian Semantic Web Workshop*, 166, 2005.
12. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.