User-oriented Description of Emerging Services in Ambient Systems

Maroun Koussaifi

User-oriented Description of Emerging Services in Ambient Systems

Maroun Koussaifi
Supervisors: Jean-Paul Arcangeli, Jean-Michel Bruel and Sylvie Trouilhet

University of Toulouse / IRIT, France

Abstract. Ambient intelligence aims at providing users the right services at the right time. Our solution composes software components and their services, automatically and on the fly, and makes composite services emerge from the environment. An important question is their intelligible presentation to an average user (not a service composition expert). Our approach consists in the automatic generation of user-oriented descriptions from unit descriptions of components and services. For that, we propose a domain-specific language for component and service descriptions and a combining method.

1 Introduction

Applications of the Internet of Things, ambient and cyber-physical systems consist of fixed or mobile connected devices. Devices host independently developed and managed software components that provide services specified by interfaces and, in turn, may require other services [8]. Components are building blocks that can be assembled by binding required and provided services to build composite applications. Due to mobility and separate management, devices and software components may appear and disappear without this dynamics being foreseen.

Humans are at the core of these dynamic and open systems. Ambient intelligence aims at offering them a personalized environment adapted to the current situation, anticipating their needs and providing them the right applications at the right time with the least effort possible.

We are currently exploring and designing a solution in which components are dynamically and automatically assembled to build new composite applications and so customize the environment at runtime. Our approach is quite disruptive: unlike the traditional goal-directed top-down mode, applications are built on the fly in bottom-up mode from the components that are present and available at the time, without user needs being made explicit, and without relying on predefined plans. That way, composite applications continuously emerge from the environment, taking advantage of opportunities as they arise: for example, a slider on a smartphone, a software adapter, and a connected lamp can opportunely be composed to provide the user with a lighting service when entering a room.

Composition is automated by an *opportunistic composition engine* (OCE), in line with the principles of autonomic computing [4]. OCE senses the existing components and proactively makes the connections. The heart of OCE is a multi-agent system where agents manage the services and their connections [10]. To make the right decisions and build relevant applications, the agents learn online and by reinforcement.

OCE behavior and decisions are out of the scope of this paper, which focuses on placing the user in control of the deployment of emerging applications [6]. First of all, she/he must be informed of an emerging, possibly unexpected, application. Then, depending on its interest, she/he must be able to accept or reject it, or possibly modify it (provided that she/he has the required skills). For that, an editor displays the component-based architecture of the application, and allows modification [6]. However, such representation is only accessible to experts in component-based programming. Moreover, it does not explain the service that is offered. It is then essential to assist the user in the appropriation of the applications pushed by the engine. For that, they must be described in a useful and understandable way. This is especially important since the user's reactions are the sources of feedback for learning: based on them, OCE builds and updates a model of the user's preferences and habits.

This paper focuses on a solution to provide the user with an intelligible description of emerging composite applications. Section 2 states the problem and the requirements; Section 3 analyzes the state of the art; Section 4 describes the solution and preliminary results; Section 5 concludes and outlines the main perspectives.

2 Problem statement and requirements

In the absence of prior specification, emerging applications may be unknown and possibly surprising for the user. Thus, the way new applications are presented is critical. The purpose of the application must be explicit **[R1-Semantics]** (e.g. "The application allows to light up the lamp"), and how to use the application must be explicit too [**R2-Usage**] (e.g. "Press the switch to turn ON/OFF the light"). The description must also be understandable [**R3-Intelligibility**]: here, we target average users that are not familiar with programming or computer science (e.g. the inhabitant of a smart house or a public transport traveller in a smart city). Moreover, the description should remain intelligible even if the application consists of one to a few dozen components [**R4-Scalability**].

Henceforth, the problem is to build and display user-oriented understandable descriptions. As applications are automatically assembled, the descriptions of the services they provide must be computed automatically from the descriptions of their components and services [**R5-Automation**]. Besides, the language that supports the description of components and services should be expressive and easy-to-use for engineers that provide them [**R6-Expressiveness**].

3 Related work

There exists many solutions for functional and extrafunctional service description. They are mainly used to support automated service discovery and selection in a top-down composition approach, that tends to build a complex service from unit ones. However, there exists no solution which aims at combining descriptions to build the description of a composite service to be presented to the user. To the best of our knowledge, there is no work that meets our requirements, mainly those concerning usage, intelligibility, and automated processing, in the context of bottom-up and goal-free application construction. In the following, we synthesize the related work.

For whom and why describing a service? Basically, service description is used as documentation for developers. It allows services to be located and used, as it is the case with WSDL. In Web service composition, the required services are specified explicitly. Then, in a more or less automatic approach [7], they are discovered and selected, based on their similarity with the expected ones, then assembled together. Hence, this consists in a top-down mode approach where the service description are no longer necessary in the composition phase. In [9], authors propose a user-centric composition platform: end-users first specify their goals using keywords, then the editor present the possible services that answer his/her needs, and suggest possible and user-changeable processes.

How to describe a service? In automated service composition approaches, the description of services varies according to the requirements of the discovery and selection steps. The different solutions for service description have been classified [3]. Descriptions may be limited to a syntactic way. For example, in object-oriented middleware (e.g. Java RMI), services are located only through a name. Otherwise, descriptions may be functional. It can have the form of signature with inputs and outputs, likely completed by preconditions and effects [5]. However, signature is not enough because their might be different functions with the same signature or even two services with the same function but with different quality levels. Therefore, a service description should include extrafunctional characteristics that is QoS-related properties. According to [3], OWL-S has become a standard for industrial service composition. OWL-S is an ontology-based language for describing semantic Web services that enables their automated discovery, composition and use. Ontology-driven description of services have proved to be efficient for selection and composition [9].

4 Proposition

Building the description of a composite application consists in combining unit descriptions of the components and their services. For that, we propose *(i)* a domain-specific description language and *(ii)* a combination method. Due to space limitation, we do not detail our solution here (see [1]). The idea is to describe both the services and the components with their services and possibly their states. Descriptions mainly rely on logical rules which state how services interact and transform data, and how the user can use the interactive components. Engineers that develop components provide component and service descriptions. In addition, the latter are completed by the engine with the emerging bindings. Then, the rules are combined to produce application-level rules.

In order to validate our approach, we have developed a prototype solution and tested it on several use cases, with different component assembly topologies. In the following, we present two of them.

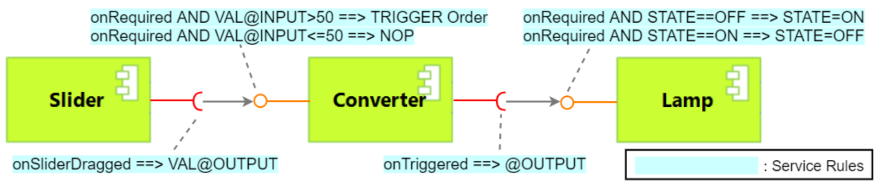

onRequired AND VAL@INPUT>50 ==> TRIGGER Order
onRequired AND VAL@INPUT<=50 ==> NOP
onRequired AND STATE==OFF ==> STATE=ON
onRequired AND STATE==ON ==> STATE=OFF
Slider
Converter
Lamp
onSliderDragged ==> VAL@OUTPUT
onTriggered ==> @OUTPUT
: Service Rules

Fig. 1. Structural representation of the lighting service

Lighting service. The application (see Fig. 1) consists of three components assembled in pipeline mode: a slider, a converter and a lamp. The slider acts as a switch. It requires the *ProcessValue* service. The converter provides the *Transform* service (that subsumes the *ProcessValue* service): it receives a value and, if greater than 50, transforms it into an command for the lamp through the *Order* required service. The lamp provides the *OnOff* service (that subsumes the *Order* service). Fig. 2 shows the rules resulting from the combination of the service rules highlighted in Fig. 1. Rules are then translated into a more intelligible version of the supplied service (see Fig. 3).

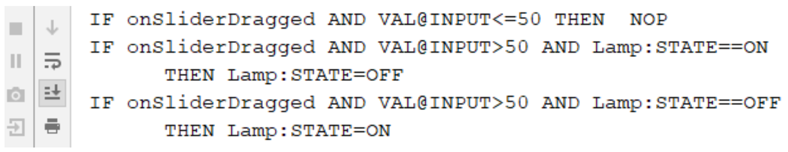

```
IF onSliderDragged AND VAL@INPUT<=50 THEN  NOP
IF onSliderDragged AND VAL@INPUT>50 AND Lamp:STATE==ON
      THEN Lamp:STATE=OFF
IF onSliderDragged AND VAL@INPUT>50 AND Lamp:STATE==OFF
      THEN Lamp:STATE=ON
```

Fig. 2. Description rules of the lighting service

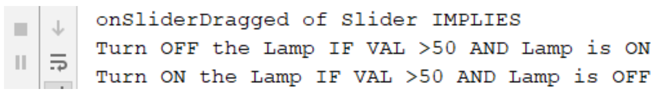

```
onSliderDragged of Slider IMPLIES
Turn OFF the Lamp IF VAL >50 AND Lamp is ON
Turn ON the Lamp IF VAL >50 AND Lamp is OFF
```

Fig. 3. User-oriented textual description of the lighting service

Multiple lighting service. The application (see Fig. 4) uses a wall switch and a component responsible of controlling two lamps at the same time, assembled in a star topology. Fig. 5 shows the rules resulting from the combination of the rules highlighted in Fig. 4. Fig. 6 shows the same rules but in a user-oriented intelligible version.

In this example, the lamps are commanded in parallel. Note that our solution supports other types of composition operators, e.g. a sequence operator.

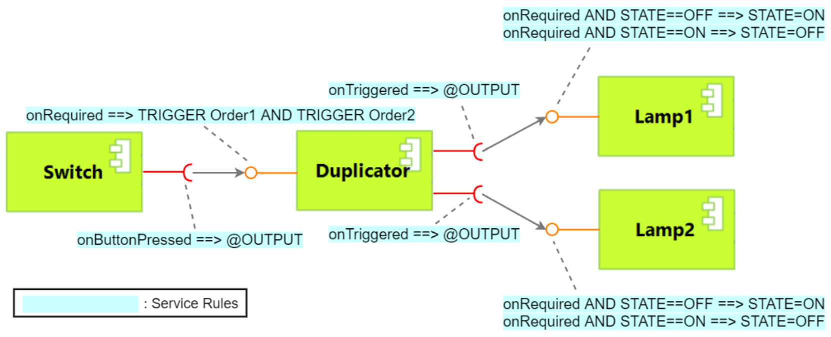
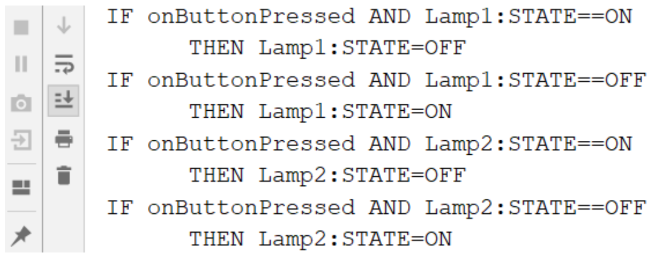
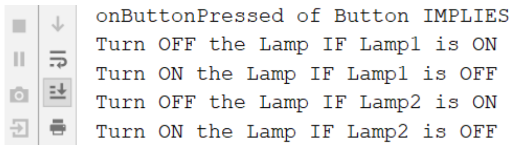

onRequired AND STATE==OFF ==> STATE=ON
onRequired AND STATE==ON ==> STATE=OFF
onTriggered ==> @OUTPUT
onRequired ==> TRIGGER Order1 AND TRIGGER Order2
Lamp1
Switch
Duplicator
Lamp2
onButtonPressed ==> @OUTPUT
onTriggered ==> @OUTPUT
: Service Rules
onRequired AND STATE==OFF ==> STATE=ON
onRequired AND STATE==ON ==> STATE=OFF

Fig. 4. Structural representation of the multiple lighting service



```
IF onButtonPressed AND Lamp1:STATE==ON
      THEN Lamp1:STATE=OFF
IF onButtonPressed AND Lamp1:STATE==OFF
      THEN Lamp1:STATE=ON
IF onButtonPressed AND Lamp2:STATE==ON
      THEN Lamp2:STATE=OFF
IF onButtonPressed AND Lamp2:STATE==OFF
      THEN Lamp2:STATE=ON
```

Fig. 5. Description rules of the multiple lighting service



```
onButtonPressed of Button IMPLIES
Turn OFF the Lamp IF Lamp1 is ON
Turn ON the Lamp IF Lamp1 is OFF
Turn OFF the Lamp IF Lamp2 is ON
Turn ON the Lamp IF Lamp2 is OFF
```

Fig. 6. User-oriented textual description of the multiple lighting service

5 Conclusion and Perspectives

In this paper, we have exposed an approach that aims to answer most of the identified requirements (see section 2): **[R5-Automation]** by automatically generating user-oriented descriptions; **[R1-Semantics]** by the description of the behaviour of the assembly by explicit rules; **[R2-Usage]** by integrating dedicated operators in the language description; and **[R3-Intelligibility]** by making the descriptions intelligible thanks to functions combination algorithms and generation of descriptions in natural language. We have experimented several use cases with standard topologies that show that our approach can meet those requirements.

At this point of our work, through user-oriented textual descriptions, average users can be informed and understand the service that is offered by emerging composite applications. Further experiments must now be carried out on more complex applications and topologies to address the missing requirement: **[R4-Scalability]**. In addition, real users should be involved in the experiments to improve and validate intelligibility and scalability of the presentation.

Our description language being a domain-specific language, and the input assembly being a model, Model-Driven Engineering (MDE) which has been proved useful in this particular case [2] will allow us to define transformation between assemblies and their descriptions. In order to easily upgrade and extend our description language, we intend to fully use the power of MDE approaches and tools to support the automatic generation of combination algorithms from the description language definition itself. In addition, using MDE to manipulate (e.g. fold/unfold) the descriptions should help to address the scalability issue [**R4-Scalability**].

Finally, we plan to investigate the use of ontologies to help in the combination process, in order to provide more intelligible descriptions (e.g. by aligning heterogeneous but related service concepts). This should limit the risk of rejection of the service by the end-user due to misdescription of emerging applications.

References

1. Component and service description language for automated description of composite applications. `https://www.researchgate.net/publication/333675107_Component_and_service_description_language_for_automated_description_of_composite_applications`, Accessed: 2019-06-10
2. Bruneliere, H., Eramo, R., Gomez, A., Besnard, V., Bruel, J.M., Gogolla, M., Kästner, A., Rutle, A.: Model-Driven Engineering for Design-Runtime Interaction in Complex Systems: Scientific Challenges and Roadmap. In: MDE@DeRun 2018 workshop. LNCS, vol. 11176 (Jun 2018). https://doi.org/10.1007/978-3-030-04771-9_40
3. Fanjiang, Y., Syu, Y., Ma, S., Kuo, J.: An overview and classification of service description approaches in automated service composition research. IEEE Transaction on Services Computing **10**(2), 176–189 (March 2017). https://doi.org/10.1109/TSC.2015.2461538
4. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (Jan 2003). https://doi.org/10.1109/MC.2003.1160055
5. Klusch, M.: Semantic web service description. In: CASCOM: Intelligent Service Coordination in the Semantic Web. pp. 31–57. Birkhäuser Basel, Basel (2008)
6. Koussaifi, M., Trouilhet, S., Arcangeli, J.P., Bruel, J.M.: Ambient intelligence users in the loop: Towards a model-driven approach. In: Mazzara, M., Ober, I., Salaün, G. (eds.) Software Technologies: Applications and Foundations. pp. 558–572. Springer (2018). https://doi.org/10.1007/978-3-030-04771-9_42
7. Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S., Xu, X.: Web services composition: A decade's overview. Information Sciences **280**, 218 – 238 (2014). https://doi.org/10.1016/j.ins.2014.04.054
8. Sommerville, I.: Component-based software engineering. In: Software Engineering, chap. 16, pp. 464–489. Pearson Education, 10 edn. (2016)
9. Xiao, H., Zou, Y., Tang, R., Ng, J., Nigul, L.: Ontology-driven service composition for end-users. Service Oriented Computing and Applications **5**(3), 159 (Mar 2011). https://doi.org/10.1007/s11761-011-0081-z
10. Younes, W., Trouilhet, S., Adreit, F., Arcangeli, J.P.: Towards an intelligent user-oriented middleware for opportunistic composition of services in ambient spaces. In: Proc. of the 5^{th} Workshop on Middleware and Applications for the Internet of Things. pp. 25–30. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3286719.3286725