Interactive Reinforcement Learning for Software Composition via Software Product Lines -Approach and Research Questions

Kévin Delcourt, Françoise Adreit, Jean-Paul Arcangeli, Sylvie Trouilhet

Institut de Recherche
en Informatique de Toulouse

UNIVERSITÉ
TOULOUSE III
PAUL SABATIER

Interactive Reinforcement Learning for Software Composition via Software Product Lines - Approach and Research Questions

IRIT/RR–2022–03–FR

Mars 2022

Kévin DELCOURT,
Françoise ADREIT,
Jean-Paul ARCANGELI,
Sylvie TROUILHET

Institut de Recherche en Informatique de Toulouse (IRIT)

Abstract

Opportunistic software composition of services is a novel interactive approach for the construction of software in open and dynamic ambient environments. The goal is to dynamically provide relevant applications to a user without predefined assembly plan or functional requirements. For that, an intelligent composition system builds, through distributed and interactive reinforcement learning, assemblies of software components present in the user's environment.

A current limit of this approach is that in some situations, for example at startup, the composition engine lacks information and as a result proposes random assemblies to the user. The contribution discussed in this paper assists the engine in such situations by adding a feature model generated from the ambient environment. Thus, the engine gathers additional knowledge comparing its proposition to this feature model, providing more pertinent assemblies to the user.

Keywords

Reinforcement Learning, Software Composition, Multi-Agent System, Software Product Line

I Introduction

Today's users are living in cyber-physical ambient environments that are more and more complex with the increasing number of devices in the Internet of Things. These environments consist of fix or mobile devices, driven by software components that provide services and in turn may require services to operate.

Those devices and components are generally developed, installed, activated, and assembled independently of each other. They may appear or disappear with unpredictable dynamics, giving to ambient environments an open and unstable nature. In such a context, applications based on component assemblies are hard to design, maintain and adapt.

In order to tackle these issues, our project aims to design and build a composition system named OCE, for Opportunistic Composition Engine, that automatically and dynamically assembles software components in order to build applications that are adapted to the current state of the environment and the user [7]. In order to build pertinent applications without explicit user needs, OCE learns from user feedback in an endless online reinforcement learning mode [5].

However, in some situations, OCE has no knowledge to base its decision on, and makes random choices. This is the case at the system start-up and when new components appear. In this paper we discuss a potential solution to this problem, involving software product lines, through the generation of feature models.

This paper is structured as follows. Section 2 presents the global architecture of this composition system. Section 3 proposes an approach to improve our solution based on software product lines. Section 4 concludes and states the current limitations and questions relative to our work.

II Interactive Reinforcement Learning for Composition

Fig. 1 (activities in gray) shows the overall workings of the composition system OCE, in relation to the user. OCE periodically senses the ambient environment in order to discover available components, connects corresponding provided and required services with each other and thereby constructs on the fly composite applications.

OCE is designed in a MAS - Multi-Agent System - architectural style [6]. Any service sensed in the pervasive environment is managed by a dedicated agent [7]. The agents cooperate in order to choose the pertinent connections to realize between the available components. An agent's decision is based on its local view of the assembly and of the environment, and its estimated values of the other agents: these values are computed from the feedback gathered throughout the previous OCE executions. They are memorised by the agents as situations already encountered (compatible services and interest of connecting with).

The task of presenting the application to the user whenever it emerges is assigned to ICE, for Interactive Control Environment. The user is therefore in the loop [1]. Using model-driven engineering, ICE aims to present emerging applications in an intelligible manner: several graphical representation styles are available, from abstract ones to concrete and technical ones. Non-expert users can choose the one that best suits their preferences [2].

Thanks to ICE, the user can either accept, reject or modify the proposed application. Feedback data are then deducted from these actions : it allows OCE to learn and build knowledge about the user related to the current situation, in an endless online reinforcement learning mode [5].

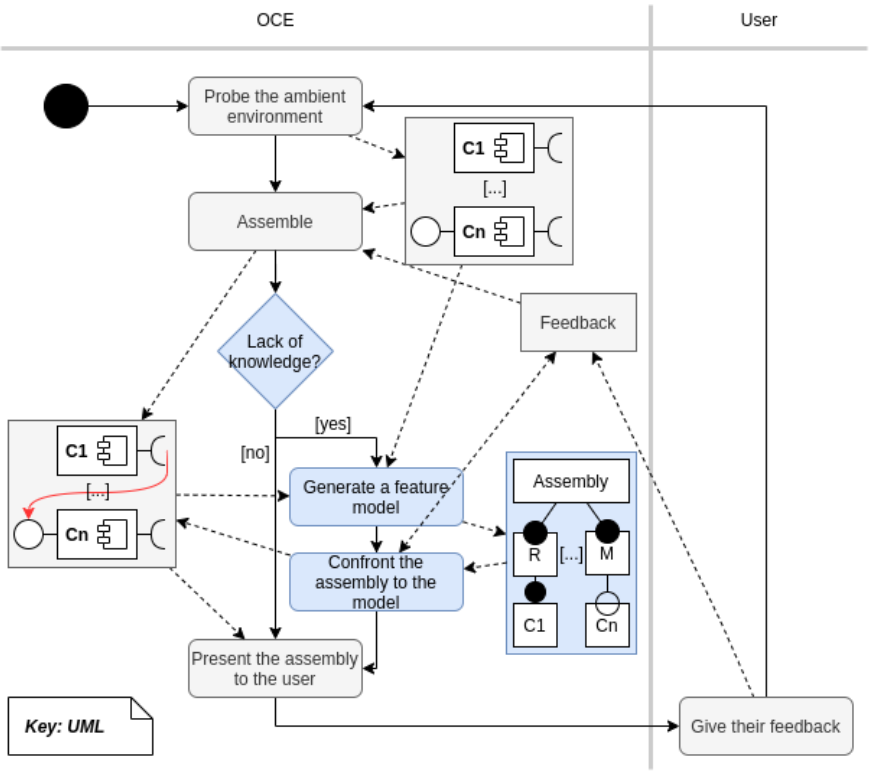

Figure 1: Composition process (in grey) augmented by the feature model (in blue)

III Involvment of Software Product Lines

The concept of Software Product Lines [4] is based on the idea of developing and managing a line of applications (i.e., applications that share some common components or features) as a whole, rather than managing each application independently.

Software Product Lines are based on feature models: a synthetic representation of all the possible products of a line. An example of feature model is available in Fig. 1: it expresses that an assembly has the features R and M, and that M may require another feature Cn. The detail of possible relationships between features is available in [3].

One limit of the composition system OCE is that in certain situations, for a new environment or at the system start-up, collected knowledge may be insufficient for the agents to decide on the best potential connection. In such a situation, without estimated values of the other agents, an agent will choose a connection randomly. And so, it is likely that the assembly proposal does not satisfy the user and will have to be modified by him.

Fig. 1 presents (in blue) our addition to the basic composition process (in grey): in order to allow OCE to gather knowledge in those situations, we generate a feature model from the environment. This model describes all the assemblies that are supposedly ”acceptable” to the user: if OCE’s proposition is not included in this model then it is rejected, thus generating helpful feedback without additional user input.

Concerning the generation of this feature model, we are at the moment considering two potential methods that we derived from the observation of our use cases : one based on the components structure (currently fully automated) and one based on their semantics (work in progress).

Early testing of this solution seems to show that it indeed produces better applications for the human user. However, the tests have so far been done on very small data sets, so we can’t yet conclude on the validity of our approach.

IV Conclusion

The idea of using software product lines to improve OCE's learning origins from the fact that assemblies produced by OCE share components or services that are re-used through time, in the same way that features are shared between products of a software product line. Feature model generation seems useful to avoid overloading the user, especially in case of lack of knowledge.

Additional tests on a great number of realistic use cases, with real-world users and components, are necessary to validate our proposition. The main challenge is obtaining this data : it is not possible to generate it manually or automatically since we want it to represent the complexity of human users. Furthermore, the determination of the method of generation of those feature models is still at an early stage, and other possibilities may be considered in the future.

However, our experiences up to this point show that the current prototypes make assembly propositions that are most of the time pertinent to the user. In situations where no knowledge of the user's preferences are available, it produces assemblies that seem to be better than randomly generated ones. In addition, it reduces the burden on the user, which we consider to be a positive result.

Bibliography

[1] Evers, C., Kniewel, R., Geihs, K., Schmidt, L.: The user in the loop: Enabling user participation for self-adaptive applications. Future Generation Computer Systems **34**, 110–123 (2014)

[2] Koussaifi, M., Arcangeli, J.P., Trouilhet, S., J.-M., B.: Model-Driven Engineering for End-Users in the Loop in Smart Ambient Systems. Technical report IRIT/RR–2020–06–FR, IRIT (September 2020)

[3] Lopez-Herrejon, R.E., Illescas, S., Egyed, A.: A systematic mapping study of information visualization for software product line engineering. Journal of software: evolution and process **30**(2), e1912 (2018)

[4] Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering: foundations, principles, and techniques, vol. 1. Springer (2005)

[5] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, 2nd edn. (2018)

[6] Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley (2009)

[7] Younes, W., Trouilhet, S., Adreit, F., Arcangeli, J.P.: Agent-mediated application emergence through reinforcement learning from user feedback. In: 29th IEEE Int. Conf. on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE Press (2020), `https://hal.archives-ouvertes.fr/hal-02895011`