A Feature-Based Context-Oriented Approach to Dynamic Software Evolution

Duhoux, Benoît; Mens, Kim; Dumas, Bruno

Publication date:
2020

A Feature-Based Context-Oriented Approach to Dynamic Software Evolution

Benoît Duhoux
ICTEAM
Université catholique de Louvain
benoit.duhoux@uclouvain.be

Kim Mens
ICTEAM
Université catholique de Louvain
kim.mens@uclouvain.be

Bruno Dumas
PReCISE / NADI
Université de Namur
bruno.dumas@uclouvain.be

***Abstract*—Context-oriented programming enables dynamic software evolution by supporting the creation of software systems that dynamically adapt their behaviour depending on the context of their surrounding environment. Conceiving such systems is complex through due to their high dynamicity and the combinatorial explosion of possible contexts and corresponding features that could be active. We present the current status of our feature-based context-oriented programming approach, illustrate how it helps software developers to conceive highly-dynamic systems, and discuss some remaining challenges of our approach.**

I. Introduction

Context-aware systems [1] are systems that are aware of their surrounding environment and adapt their behaviour at runtime depending on contextual changes in that environment. Context-oriented programming [2] is a particular programming methodology to build such systems. Upon sensing a new particular situation in the surrounding environment, which is reified as a context, the system activates this context and then continues by selecting and activating fine-grained features corresponding to that context. These features, representing functionalities specific to that context, are then installed in the system to refine its behaviour at runtime [3]. Such systems thus evolve dynamically depending on the contexts describing their surrounding environment. Designing and building such systems using context-oriented programming remains quite complex due to their high variability and dynamicity. We present our particular feature-based context-oriented programming approach and illustrate how it can help software developers to design and implement such highly dynamic systems.

II. A feature-based context-oriented approach

Designing feature-based context-oriented systems can be quite complex because the designers have to think about what features such a system should have and to what contexts it should adapt their behaviour. Adapted components can range from user interface components to core logic or even data components. Designers should take into account which contexts trigger which features. They also need to model and respect dependencies and constraints between the contexts, and between features. Designing and developing such programs thus remains a quite challenging task.

In order to help designers and developers of context-aware systems in their complex task, over the past few years we have been developing and refining a feature-based context-oriented software development approach.

The first aspect we tackled concerns the foundations of the approach, i.e., how systems are conceived in terms of a context model, a feature model and the mapping between them. In more traditional context-oriented programming languages, contexts and their behavioural adaptations are more strongly linked than in our approach which tries to make a clear separation between them, as inspired by Hartmann and Trew [4]. This separation makes it easier to model the contexts that reify situations in the surrounding environment, the features that implement the behavioural adaptations, and what features are triggered by what contexts.

We then proposed an actual implementation of such a feature-based context-oriented programming language [5], two visualisations tools to better inspect some key language concepts [6], [7], and conducted several case studies to validate the language concepts. Finally, we worked on an incremental software development methodology to help stakeholders conceive such highly dynamic systems.

Despite our current progress, several important issues remain to be explored such as the data concern or testing. How can we conceive or query databases in such a context-oriented fashion? How can we test such systems to ensure code quality, coverage and predictability? Finally, how can we help end-users to accept and adopt such dynamic evolutive systems?

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and Ubiquitous Computing*. Springer, 1999, pp. 304–307.

[2] R. Hirschfeld, P. Costanza, and O. Nierstrasz, "Context-oriented programming," *Journal of Object Technology*, vol. 7, no. 3, pp. 125–151, 2008.

[3] K. Mens, N. Cardozo, and B. Duhoux, "A context-oriented software architecture," in *COP '16*. ACM, 2016, pp. 7–12.

[4] H. Hartmann and T. Trew, "Using feature diagrams with context variability to model multiple product lines for software supply chains," in *SPLC '08*. IEEE, 2008, pp. 12–21.

[5] B. Duhoux, K. Mens, and B. Dumas, "Implementation of a feature-based context-oriented programming language," in *COP '19*. ACM, 2019, pp. 9–16.

[6] B. Duhoux, B. Dumas, H. S. Leung, and K. Mens, "Dynamic visualisation of features and contexts for context-oriented programmers," in *EICS '19*. ACM, 2019.

[7] B. Duhoux, K. Mens, and B. Dumas, "Feature visualiser: An inspection tool for context-oriented programmers," in *COP '18*. ACM, 2018, pp. 15–22.