RubyCOP

Duhoux, Benoît; Mens, Kim; Dumas, Bruno

Published in:
BENEVOL 2022

Publication date:
2022

RubyCOP: A Feature-Based Context-Oriented Programming Framework

Presentation · September 2022

DOI: 10.13140/RG.2.2.15156.53127

3 authors:

Benoît Duhoux
Université Catholique de Louvain - UCLouvain

Kim Mens
Université Catholique de Louvain - UCLouvain

Bruno Dumas
University of Namur

RubyCOP: A Feature-Based Context-Oriented Programming Framework

Benoît Duhoux
INGI / ICTEAM / UCLouvain
benoit.duhoux@uclouvain.be

Kim Mens
INGI / ICTEAM / UCLouvain
kim.mens@uclouvain.be

Bruno Dumas
PReCISE / NADI / UNamur
bruno.dumas@uclouvain.be

I. Introduction

Context-aware systems [1] are systems that adapt their behavior dynamically when sensing changes in the surrounding environment in which they run. These changes may come from changing user preferences, external sensors (weather, localisation), or internal sensors (battery, memory) of the device. The paradigm of context-oriented programming [2], [3] provides dedicated programming language abstractions to adapt the behaviour of a software system dynamically upon changing contexts. In context-oriented programming (COP), contexts and behavioural adaptations are first-class language entities. The behavioural adaptations get (de)activated in the code whenever their corresponding contexts become (de)activated. Many different COP languages exist [4]–[14]. We also proposed our own specific programming framework [15], RubyCOP, to implement context-oriented systems in which we explicitly separate the notions of contexts and features. This clear separation promotes a better maintainability and reusability in developping such systems. Our RubyCOP programming framework is part of a more complete approach to feature-based context-oriented software development, which also consists of an architecture [16], a supporting development methodology and two visualisation tools [17], [18]. This presentation will focus only on the programming framework.

II. Feature-based context-oriented programming

Feature-based context-oriented programming (FBCOP) builds upon context-oriented programming [3], feature modelling [19] and dynamic software product lines [20]. In this new programming paradigm, contexts and features are clearly separated and modelled in terms of a feature diagram to represent, respectively, a context model and feature model. Such models are tree-like structures where the nodes represent the contexts or features and the edges represent the constraints between the different nodes. A context-feature mapping then expresses what contexts trigger what features in order to adapt the system behaviour when contexts change.

III. RubyCOP

Following the underlying principles of FBCOP, we built an application programming framework on top of the *Ruby* programming language. This programming framework provides native buildings blocks to declare contexts and features as first-class citizens. It also provides dedicated abstractions and language constructs to define the context model, the feature model and the mapping between the context and feature model. In addition it offers specific language constructs to implement the adaptive behaviour of the features (code of the features that adapts or refines the system behaviour). Some of these dedicated language constructs are listed in Table I.

TABLE I
RubyCOP's Language Constructs for Feature Definitions.

can_adapt	Declares what application classes may be adapted by a given feature part.
set_prologue	Defines what code of a feature part should be executed automatically after the activation of a feature.
set_epilogue	Defines what code of a feature part should be executed automatically before the deactivation of a feature.
proceed	Calls the previous adaptation or default behaviour of a given method.

Finally, our programming framework also abstracts the entire process from the (de)activation of contexts to the deployment of features in the system behaviour, via the selection of features and their (de)activation. This includes the attempt to (de)activate the contexts and features according to the constraints imposed by their corresponding models, and their commits or rollbacks if the models are valid or not (i.e., if all model constraints are satisfied).

IV. Assessing the framework with programmers

To assess whether our programming framework is understandable, useful and usable, we conducted two user studies with programmers. It is important to assess such properties with developers because a too complex or unusable framework would never be used in the future to conceive such highly dynamic systems. In the presentation we will discuss a first user study with 41 students who played the role of programmers of FBCOP applications.

While they appreciated the framework's expressiveness to build context and feature models, they lacked some expressiveness to implement feature definitions. They also found the framework to be complex, despite of the supporting methodology and visualisation tools. This could be explained by the steep learning curve of the full FBCOP approach and a lack of documentation of the framework. But also by the intrinsic complexity of building context-aware systems that can adapt their behaviour dynamically to many different contexts.

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*, H.-W. Gellersen, Ed. Berlin, Heidelberg: Springer, 1999, pp. 304–307.

[2] R. Keays and A. Rakotonirainy, "Context-oriented programming," in *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, ser. MobiDe '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 916.

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

[4] P. Costanza and R. Hirschfeld, "Language constructs for context-oriented programming: An overview of contextl," in *Proceedings of the 2005 Symposium on Dynamic Languages*, ser. DLS '05. New York, NY, USA: ACM, 2005, p. 110.

[5] S. González, K. Mens, and A. Cádiz, "Context-oriented programming with the ambient object system," *Journal of Universal Computer Science*, vol. 14, no. 20, pp. 3307–3332, nov 2008.

[6] R. Hirschfeld, P. Costanza, and M. Haupt, "Generative and transformational techniques in software engineering ii." Springer, 2008, ch. An Introduction to Context-Oriented Programming with ContextS, pp. 396–407.

[7] M. Appeltauer, R. Hirschfeld, and T. Rho, "Dedicated programming support for context-aware ubiquitous applications," in *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. IEEE, sep 2008, pp. 38–43.

[8] J. Lincke, M. Appeltauer, B. Steinert, and R. Hirschfeld, "An open implementation for context-oriented layer composition in contextjs," *Science of Computer Programming*, vol. 76, no. 12, pp. 1194 – 1209, 2011, special Issue on Software Evolution, Adaptability and Variability.

[9] T. Kamina, T. Aotani, and H. Masuhara, "Eventcj: A context-oriented programming language with declarative event-based context transition," in *Proceedings of the Tenth International Conference on Aspect-Oriented Software Development*, ser. AOSD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 253264.

[10] S. González, N. Cardozo, K. Mens, A. Cádiz, J.-C. Libbrecht, and J. Goffaux, *Subjective-C*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 246–265.

[11] T. Aotani, T. Kamina, and H. Masuhara, "Featherweight eventcj: A core calculus for a context-oriented language with event-based per-instance layer transition," in *Proceedings of the 3rd International Workshop on Context-Oriented Programming*, ser. COP '11. ACM, 2011, pp. 1:1–1:7.

[12] G. Salvaneschi, C. Ghezzi, and M. Pradella, "Contexterlang: Introducing context-oriented programming in the actor model," in *Proceedings of the 11th Annual International Conference on Aspect-oriented Software Development*, ser. AOSD '12. New York, NY, USA: ACM, 2012, pp. 191–202.

[13] T. Poncelet and L. Vigneron, "The Phenomenal Gem: Putting features as a service on Rails," Master's thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, Jun. 2012.

[14] S. González, K. Mens, M. Colacioiu, and W. Cazzola, "Context traits: Dynamic behaviour adaptation through run-time trait recomposition," in *Proceedings of the 12th Annual International Conference on Aspect-oriented Software Development*, ser. AOSD '13. New York, NY, USA: ACM, 2013, pp. 209–220.

[15] B. Duhoux, K. Mens, and B. Dumas, "Implementation of a feature-based context-oriented programming language," in *Proceedings of the Workshop on Context-Oriented Programming*, ser. COP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 916.

[16] K. Mens, N. Cardozo, and B. Duhoux, "A context-oriented software architecture," in *Proceedings of the 8th International Workshop on Context-Oriented Programming*, ser. COP '16. New York, NY, USA: ACM, 2016, pp. 7–12.

[17] B. Duhoux, K. Mens, and B. Dumas, "Feature visualiser: An inspection tool for context-oriented programmers," in *Proceedings of the 10th International Workshop on Context-Oriented Programming: Advanced Modularity for Run-Time Composition*, ser. COP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1522.

[18] B. Duhoux, B. Dumas, H. S. Leung, and K. Mens, "Dynamic visualisation of features and contexts for context-oriented programmers," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '19. New York, NY, USA: Association for Computing Machinery, 2019.

[19] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Carnegie-Mellon University Software Engineering Institute, Tech. Rep., November 1990.

[20] H. Hartmann and T. Trew, "Using feature diagrams with context variability to model multiple product lines for software supply chains," in *Proceedings of 12th International Software Product Line Conference*, ser. SPLC '08. IEEE, 2008, pp. 12–21.