Louvain School of Engineering (EPL)
Computing Science Engineering Department (INGI)
Reflection and Context-Oriented Programming (R+COP)
Course assignment

rev. 2012-07-16

COP Framework Extensions

During the practical sessions, you developed a basic Context-Oriented Programming (COP) framework. This framework contains the minimum functionality that is necessary for COP to be of any practical use, namely:

Lab4 The necessary infrastructure to work with first-class contexts.

Lab5 A means to define behavioural adaptations for different contexts.

Lab6 A means to invoke overridden behaviour within each adaptation.

Lab7 A means to choose among available adaptations when multiple contexts are active.

All in all, this permits the definition and composition of context-oriented behaviour, and however simple, it already constitutes a fully functional COP engine.

This document describes possible extensions of the framework developed so far. The proposed ideas are not step-by-step recipes that you can follow systematically, as was the case of the guided practical sessions. Rather, the extensions are formulated in a general way, leaving you room to narrow the actual problem you want to tackle, and to provide your own creative solution.

Extensions

1. *Context-Aware Blocks*

   This extension invites you to explore the interaction between contexts and code blocks. What does it mean to have context-aware blocks? How to define them?

   For the design of context-aware blocks, you should consider the paper by Clarke et al. [1], and draw inspiration for your own design.

2. *Context Dependencies* Contexts are often interrelated to each other. The (de)activation of a context sometimes entails the corresponding (de)activation of other contexts. Such dependencies could be expressed explicitly by programmers.

   In this extension, you are invited to explore context dependencies, by drawing inspiration from Subjective-C [3].

3. *Context Combinations* The basic COP framework allows the definition of methods specialised on exactly one context. However, it is often useful to define methods specialised on more than one particular context.

   In this extension, you should define a mechanism that makes it possible to define such multiply-specialised methods. You can draw inspiration from context combinations as defined in Ambience [2, 4].

References

[1] Dave Clarke, Pascal Costanza, and Éric Tanter. How should context-escaping closures proceed? In *International Workshop on Context-Oriented Programming*, COP'09, pages 1–6. ACM Press, 2009. ISBN 978-1-60558-538-3. DOI 10.1145/1562112.1562113.

[2] Sebastián González, Kim Mens, and Alfredo Cádiz. Context-Oriented Programming with the Ambient Object System. *Journal of Universal Computer Science*, 14(20):3307–3332, 2008. ISSN 0948-6968. DOI 10.3217/jucs-014-20-3307.

[3] Sebastián González, Nicolás Cardozo, Kim Mens, Alfredo Cádiz, Jean-Christophe Libbrecht, and Julien Goffaux. Subjective-C: Bringing context to mobile platform programming. In *Proceedings of the International Conference on Software Language Engineering*, volume 6563 of *Lecture Notes in Computer Science*, pages 246–265. Springer-Verlag, 2011. ISBN 978-3-642-19439-9. DOI 10.1007/978-3-642-19440-5_15.

[4] Sebastián González, Kim Mens, and Patrick Heymans. Highly dynamic behaviour adaptability through prototypes with subjective multimethods. In *Proceedings of the Dynamic Languages Symposium*, pages 77–88, New York, NY, USA, October 2007. ACM Press. ISBN 978-1-59593-868-8. DOI 10.1145/1297081.1297094. Co-located with OOPSLA'07.