Title: Flexible Modules for Dynamic And Live Programming Languages

Keywords: compilation and language design

Laboratoire, institution et université: INRIA Lille Nord Europe

Location: Lille, France

Team: Equipe RMoD – INRIA Lille Nord Europe

Internship supervisor:

Stéphane Ducasse stephane.ducasse@inria.fr

http://stephane.ducasse.free.fr

Context

Typically a package declares the other packages it uses. We distinguish modules from packages in the sense that a package does not automatically includes a visibility mechanism (often implemented as a namespace).

While this is the normal way to think about packages, there are alternatives approaches as the one of Units in DrScheme [Unit] and now Newspeak developed by Gilad Bracha (who was responsible of the Java specification during more than 10 years) [Bracha 10]. In addition, Java new version proposes a notion of modules. With Jigsaw, Java proposed a module system on top of existing abstractions [Jigsaw] leading to an even more complex languages. Python has a slightly different semantics that may led to subtle bugs.

Subsystems was a proposal to design a somehow traditional module system for Smalltalk [Subsystem]. [Berg05] Bergel et al offer an overview of advanced module system but do not propose a concrete, scalable module system for a powerful dynamically typed language.

In a live and immersive programming language, developers modify live object representing the program itself. The semantics of a module cannot force developers to follow certain interaction paths that may break their concentration flow. It is important to support incoherent states (for example that the superclass of a class is unknown).

The other challenge is how to scope class extensions (in languages such objective-C, Pharo, ruby) it is possible to define methods defined in a different package that the class they belong to [Polito 17b]).This scoping implies that the developers should be aware of the extensions of classes in the recursive import. Defining a simple yet powerful model is still a challenge [Polito 17b].

We developed Metalo a module system to support dynamic live programming operations such as changing the superclass of a class to a non-existing class, importing a class to be defined in a module. Still Metalo is incomplete because it does not support well classes extensions.

Objectives :

The topic of this internship is to

revisit the design of Metalo to access if the live programming operations are covered.

revisit the design of Undefined classes [Polito 17]

design an extension of Metalo to support class extensions.

Personal skills

Curiosité

Envie d'apprendre

Bonne communication

Bon programmeur

Anglais

References

[Unit] Matthew Flatt and Matthias Felleisen. Units: Cool modules for hot languages. In Proceedings of PLDI '98 Conference on Programming Language Design and Implementation, pages 236–248. ACM Press, 1998.
[Berg05] Alexandre Bergel, Stéphane Ducasse and Oscar Nierstrasz, Analyzing Module Diversity, In Journal of Universal Computer Science, November,Volume 11, pp. 1613-1644, 2005, SCI impact factor 0.337
[Subsystem] Allen Wirfs-Brock, Subsystems --- Proposal, OOPSLA 1996 Extending Smalltalk Workshop, 1996.
[Polito 17] G. Polito and St\'ephane Ducasse and Luc Fabresse, First-Class Undefined Classes for Pharo, Proceedings of the 12th Edition of the International Workshop on Smalltalk Technologies 2017.

[Polito 17b] G. Polito, Camille Teruel, Stéphane Ducasse, Luc Fabresse, Scoped Extension Methods in Dynamically-Typed Languages, The art science and Engineering of Programming, 2017.

[Polito 15] G. Polito. Virtualization Support for Application Runtime Virtualization and Extension. Ph.D. Thesis 2015. Co-delivrée par l'Université de Lille et l'École des Mines de Douai.

[Polito 2014] G. Polito, S. Ducasse, L. Fabresse, N. Bouraqadi, B. Van Ryseghem: Bootstrapping reflective systems: The case of Pharo. In Science of Computer Programing 96: 141-155 (2014)

[Bracha 10] Gilad Bracha, Peter von der Ah´e, Vassili Bykov, Yaron Kashai, William Maddox, and Eliot Miranda, Modules as Objects in Newspeak, ECOOP 2010.

[Jigsaw 17] Jigsaw http://cr.openjdk.java.net/~mr/jigsaw/spec/ 2017