PHANtom: An Aspect Language for Pharo Smalltalk

Johan Fabry *

PLEIAD Laboratory
Computer Science Department (DCC)
University of Chile
http://pleiad.cl

Abstract

Aspect languages for Smalltalk have not kept up with advances in aspect language research. Arguably the only well-known aspect language for Smalltalk is AspectS. It is a ground-breaking contribution, especially regarding dynamic aspects, yet it lacks amenities which aspect language users have come to rely on, e.g. the use of patterns in pointcuts and the ability to declare aspect precedence. Alternative aspect languages for Smalltalk are effectively absent. As a result, currently Smalltalk lacks a modern and powerful aspect language.To address this deficit, we elected to design and build PHANtom: a modern aspect language for Pharo Smalltalk. PHANtom is designed to be an aspect language in the spirit of Smalltalk: dynamic, simple and powerful. PHANtom is a modern aspect language because it incorporates what we consider to be the best features of languages that precede it, includes recent research results in aspect interactions and reentrancy control, and is designed from the onset to be optimized and compiled where possible. This demo presents PHANtom by first providing an introduction to the language, detailing its philosophy and fundamental features. It second discusses the salient features of the language by demonstrating the use of PHANtom in an example application.

Categories and Subject Descriptors D.3.3 [*Programming Languages*]: Language Constructs and Features

General Terms Languages, Design

Keywords PHANtom, Smalltalk, AOP

* Partially funded by FONDECYT project 1090083.

AOSD'12, March 25-30, 2012, Potsdam, Germany.

Description of the Demonstration

Aspect languages for Smalltalk have not kept up with advances in aspect language research. Arguably the only well-known aspect language for Smalltalk is AspectS [5]. It dates from 2003 and, to the best of our knowledge, has not been updated significantly since then. While a ground-breaking contribution, especially regarding dynamic aspects, it lacks amenities which aspect language users have come to rely on, e.g. the use of patterns in pointcuts and the ability to declare aspect precedence. We have not encountered alternative aspect languages for Smalltalk that are intended to be general-purpose, which have been published or even well-described. For our latest experiments with AOP we wished to use Pharo Smalltalk, which meant we also required an aspect language that incorporates recent AOP research results, is solid, powerful and extensible. Current aspect languages for Smalltalk did not meet our criteria and hence we set out to build our own language, called PHANtom [4].

PHANtom is designed to be an aspect language in the spirit of Smalltalk: dynamic, simple and powerful. PHANtom is a modern aspect language because it incorporates what we deem are the best features of languages that precede it, includes recent research results, and is designed from the onset to be optimized and compiled where possible. Notable language features are:

- the use of patterns in pointcuts, taken from AspectJ [6],
- presence of inter-type declarations, taken from AspectJ,
- static precedence-based aspect ordering scheme, taken from AspectJ,
- run-time ability of advice to alter the execution of scheduled advice at the current join point, taken from Dynamic AspectJ [1],
- dynamic deployment and undeployment, from AspectS,
- all language constructs are first-class objects, taken from AspectS and AspectScheme [3],
- a symmetric view of classes and aspects, where advice are methods, taken from Eos-U [8].
- reentrancy control based on computational membranes [10]

PHANtom adds to the state of the art in aspect languages as its joining of the above language features is new. Lastly it proposes, what is to the best of our knowledge, the first granularity refinement to the static AspectJ aspect ordering scheme by allowing pointcuts to determine a local aspect ordering scheme. PHANtom improves on existing general-purpose aspect languages in Smalltalk by allowing the use of patterns for pointcut definitions, having constructs for advice execution ordering and reentrancy control. We therefore consider that PHANtom is of interest to the Smalltalk programmer wanting to use AOP, or to the AOSD conference attendee that wishes to write aspects in Smalltalk.

The demonstration will start with an introduction to PHANtom, detailing the philosophy behind the language and explaining its fundamental features before beginning with the actual demo itself. The demo itself will consist of developing an example application, where we highlight and discuss different salient features of the language in the process of performing the development activity.

Beyond AspectS, we have not been able to find significant related work on aspect languages in Smalltalk. except for a new aspect language called PHASE [7]. The goal of PHASE is however radically different from PHANTom: it is designed and implemented as a validation of the concept of meta join point model. This model defines new join points to characterize the structure and behavior of aspects, together with the corresponding pointcut language predicates. The ultimate goal of the work is to use aspects to compose aspects, since aspect composition can be considered a cross-cutting concern. To the best of our knowledge, PHASE has only been used as a proof of concept, which is in accordance to its stated goal. PHANtom on the other hand aims to be a general-purpose aspect language for Pharo Smalltalk, and already has been successfully used in various (small-sized) applications and research projects.

PHANtom is currently implemented using MethodWrappers [2], a meta-object protocol that allows the implementation of a method to be swapped with an alternative implementation. Pointcuts are parsed using the PetitParser [9] framework and are matched using the descriptions of classes and of methods. Matches identify methods that need to be replaced by the infrastructure of PHANtom that is in charge of executing advice, along with the original behavior of the method. Work is in progress to replace the current implementation of PHANtom with a complier, based on the new compiler framework (Opal) being implemented for Pharo.

Bio of the Presenter

Johan Fabry is an assistant professor in the PLEIAD laboratory of the computer science department of the University of Chile. His main research interests are the use of AOSD in building distributed systems, the design and implementation of domain-specific aspect languages and aspect composition and interaction. Further research interests include the design of pointcut languages and weaver implementations. He was co-organizer of the DSAL Workshops at GPCE'06, AOSD'07, '08 '09, '10 and '12, co-organizer of the workshops on Aspects, Dependencies, and Interactions at ECOOP'06, '07, and '08 and editor of the special section "Dependencies and Interactions With Aspects" of the journal Transactions in Aspect-Oriented Software Development.

Additional Information

Additional information on PHANtom is available on its website: http://pleiad.cl/phantom

References

[1] Ali Assaf and Jacques Noyé. Dynamic aspectj. In *Proceedings of the 2008 symposium on Dynamic languages*, DLS '08, pages 8:1–8:12, New York, NY, USA, 2008. ACM.

[2] John Brant, Brian Foote, Ralph Johnson, and Donald Roberts. Wrappers to the rescue. In Eric Jul, editor, *ECOOP'98 - Object-Oriented Programming*, volume 1445 of *Lecture Notes in Computer Science*, pages 396–417. Springer Berlin / Heidelberg, 1998.

[3] Christopher Dutchyn, David B. Tucker, and Shriram Krishnamurthi. Semantics and scoping of aspects in higher-order languages. *Science of Computer Programming*, 63(3):207 – 239, 2006. Special issue on foundations of aspect-oriented programming.

[4] Johan Fabry and Daniel Galdames. PHANtom: a modern aspect language for Pharo Smalltalk. In *Proceedings of 2011 International Workshop on Smalltalk Technologies (IWST'11)*. ACM Press, 2011.

[5] Robert Hirschfeld. AspectS - Aspect-Oriented Programming with Squeak. In Mehmet Aksit, Mira Mezini, and Rainer Unland, editors, *Objects, Components, Architectures, Services, and Applications for a Networked World*, volume 2591 of *Lecture Notes in Computer Science*, pages 216–232. Springer Berlin / Heidelberg, 2003.

[6] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William Griswold. An overview of aspectj. In Jorgen Knudsen, editor, *ECOOP 2001 - Object-Oriented Programming*, volume 2072 of *Lecture Notes in Computer Science*, pages 327–354. Springer Berlin / Heidelberg, 2001.

[7] Antoine Marot. *Preserving the Separation of Concerns while Composing Aspects with Reflective AOP*. PhD thesis, Université Libre de Bruxelles, 2011.

[8] H. Rajan and K.J. Sullivan. Classpects: unifying aspect- and object-oriented language design. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 59 – 68, may 2005.

[9] Lukas Renggli, Stéphane Ducasse, Tudor Gîrba, and Oscar Nierstrasz. Practical dynamic grammars for dynamic languages. In *4th Workshop on Dynamic Languages and Applications (DYLA 2010)*, Malaga, Spain, June 2010.

[10] Éric Tanter, Nicolas Tabareau, and Rémi Douence. Taming aspects with membranes. In *Proceedings of the 11th Workshop on Foundations of Aspect-Oriented Languages (FOAL 2012)*, Potsdam, Germany, March 2012. ACM Press.