Sub-method, partial behavioral reflection with Reflectivity

Steven Costiou[1], Vincent Aranega[2], and Marcus Denker*[1]

[1]CRIStAL – Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France – France

[2]CRIStAL – Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France – France

Résumé

This talk was given by Marcus Denker at the conference in March 2021 (https://2021.programming-conference.org/details/programming-2021-papers/5/Sub-method-partial-behavioral-reflection-with-Reflectivity-Looking-back-on-10-years)
Refining or altering existing behavior is the daily work of every developer, but that cannot be always anticipated, and software sometimes cannot be stopped. In such cases, unanticipated adaptation of running systems is of interest for many scenarios, ranging from functional upgrades to on-the-fly debugging or monitoring of critical applications.

A way of altering software at run time is using behavioral reflection, which is particularly well- suited for unanticipated adaptation of real-world systems. Partial behavioral reflection is not a new idea, and for years many efforts have been made to propose a practical way of expressing it. All these efforts resulted in practical solutions, but which introduced a semantic gap between the code that requires adaptation and the expression of the partial behavior. For example, in Aspect-Oriented Programming, a pointcut description is expressed in another language, which introduces a new distance between the behavior expression (the Advice) and the source code in itself.

Ten years ago, the idea of closing the gap between the code and the expression of the partial behavior led to the implementation of the Reflectivity framework. Using Reflectivity, developers annotate Ab- stract Syntax Tree (AST) nodes with meta-behavior which is taken into account by the compiler to produce behavioral variations. In this paper, we present Reflectivity, its API, its implementation and its usage in Pharo. We reflect on ten years of use of Reflectivity, and show how it has been used as a basic building block of many innovative ideas.

Reflectivity brings a practical way of working at the AST level, which is a high-level representation of the source code manipulated by software developers. It enables a powerful way of dynamically add and modify behavior. Reflectivity is also a flexible mean to bridge the gap between the expression of the meta-behavior and the source code. This ability to apply unanticipated adaptation and to provide behavioral reflection led to many experiments and projects during this last decade by external users. Existing work use Reflectivity to implement reflective libraries or languages extensions, featherweight code instrumentation, dynamic software update, debugging tools and visualization and software analysis tools.

*Intervenant

Reflectivity is actively used in research projects. During the past ten years, it served as a support, either for implementation or as a fundamental base, for many research work including PhD theses, conference, journal and workshop papers. Reflectivity is now an important library of the Pharo language, and is integrated at the heart of the platform.
Reflectivity exposes powerful abstractions to deal with partial behavioral adaptation, while providing a mature framework for unanticipated, non-intrusive and partial behavioral reflection based on AST annotation. Furthermore, even if Reflectivity found its home inside Pharo, it is not a pure Smalltalk-oriented solution. As validation over the practical use of Reflectivity in dynamic object-oriented languages, the API has been ported to Python. Finally, the AST annotation feature of Reflectivity opens new experimentation opportunities about the control that developers could gain on the behavior of their own software.