EVF: An Extensible and Expressive Visitor Framework for Programming Language Reuse (Artifact)*

Weixin Zhang[1] and Bruno C. d. S. Oliveira[2]

1 **The University of Hong Kong, Hong Kong, China**
wxzhang2@cs.hku.hk

2 **The University of Hong Kong, Hong Kong, China**
bruno@cs.hku.hk

Abstract

This artifact is based on **EVF**, an extensible and expressive Java VISITOR framework. **EVF** aims at reducing the effort involved in creation and reuse of programming languages. **EVF** an annotation processor that automatically generate boilerplate ASTs and AST for a given an Object Algebra interface. This artifact contains source code of the case study on "Types and Programming Languages", illustrating how effective **EVF** is in modularizing programming languages. There is also a microbenchmark in the artifact that shows that **EVF** has reasonable performance with respect to traditional visitors.

1998 ACM Subject Classification D.1.5 Object-oriented Programming, D.3.3 Language Constructs and Features, D.3.4 Processors

Keywords and phrases visitor pattern, object algebras, modularity, domain-specific languages

Digital Object Identifier 10.4230/DARTS.3.2.10

Related Article Weixin Zhang and Bruno C. d. S. Oliveira, "EVF: An Extensible and Expressive Visitor Framework for Programming Language Reuse", in Proceedings of the 31st European Conference on Object-Oriented Programming (ECOOP 2017), LIPIcs, Vol. 74, pp. 29:1–29:32, 2017.
http://dx.doi.org/10.4230/LIPIcs.ECOOP.2017.29

Related Conference European Conference on Object-Oriented Programming (ECOOP 2017), June 18-23, 2017, Barcelona, Spain

1 Scope

The artifact is designed to support repeatability of all the experiments of the companion paper, allowing users to test the framework on a variety of benchmarks. It includes **EVF**, an extensible and expressive Java VISITOR framework that aims at reducing the effort involved in creating and reusing programming languages. **EVF** is best used with an IDE like Eclipse, which automatically generates boilerplate ASTs and AST traversals for an annotated standard Object Algebra interface by saving. This artifact also contains the source code of the case study on "Types and Programming Languages" and the microbenchmark that we discussed in the companion paper.

* This work has been sponsored by the Hong Kong Research Grant Council projects number 27200514 and 17258816.

2 Content

The artifact package includes:

- *VisitProcessor*: the Java annotation processor;
- *tapl*: the case study on "Types and programming languages";
- *benchmark*: the microbenchmark.

Detailed instructions for using the artifact and for rebuilding it from scratch, provided as an `README.md` file.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of our code is available on GitHub: `https://github.com/wxzh/EVF`.

4 Tested platforms

The artifact is known to work on any platform running JDK (version 1.8 or later) with an Eclipse IDE (version 4.5.1 or later). To run the scripts, the artifact additionally requires ruby (version 2.0.0 or later) and cloc (version 1.62 or later) installed.

5 License

BSD

6 MD5 sum of the artifact

afe518a203489ef8fc1f7c93435dd35e

7 Size of the artifact

698KB

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments and suggestions.