Introduction to IWST 2022: a word from the editors

Loïc, Lagadec[1,2,3], Vincent Aranega[4,5]

[1] *Ensta-bretagne, France*
[2] *Université de Bretagne Occidentale, France*
[3] *UMR 6285 Lab-STICC*
[4] *Université de Lille 1, France*
[5] *INRIA, équipe RMoD*

Abstract

The International Smalltalk Technologies Workshop (IWST) is a forum around advances or experience in Smalltalk, bringing together Smalltalk practitioners since 2009. The IWST aims to stimulate discussion and exchange of ideas on all aspects of Smalltalk, both theoretical and practical. IWST is a co-located event with the annual European Smalltalk User Group (ESUG) conference. The 2022 edition of IWST was held in Novi Sad, Serbia, July 24-26, with Lam Research Corporation as a sponsor.

Keywords

Smalltalk, ESUG, IWST

IWST: International Workshop on Smalltalk Technologies' 22, August 24–26, 2022, Novi Sad, Serbia
✉ loic.lagadec@ensta-bretagne.fr (L. Lagadec); vincent.aranega@univ-lille1.fr (V. Aranega)
🌐 https://www.ensta-bretagne.fr/lagadec/ (L. Lagadec); https://aranega.github.io (V. Aranega)
iD 0000-0003-3778-3144 (L. Lagadec); 0000-0003-4465-1289 (V. Aranega)

CEUR Workshop Proceedings (CEUR-WS.org)

1. Scope

This report gathers six articles grouped into 3 categories of two papers each. The first one is dedicated to performance: providing guidelines to achieve high performance, measuring performance in AI. The second one deals with implementation and debugging in smalltalk through two innovative axes: model-based delta-programming, and time-travelling debuggers. The last category is dedicated to refactoring: transformation-based refactoring, and automatic generation of class comments.

2. Accepted papers

- **How Fast is AI in Pharo? Benchmarking Linear Regression (Cat. 1):**
  Like many other modern programming languages, Pharo is extending its applications to computationally demanding areas such as machine learning, big data, crypto-currencies, etc. This raises a need for fast numerical computation libraries. In this work, the authors propose to speed up low-level computations by calling routines from highly optimized external libraries, e.g., LAPACK or BLAS, via the Foreign Function Interface (FFI).
- **Design Principles for a High-performance Smalltalk (Cat. 1):**
  Smalltalk and related languages have been the subject of many implementations, with very different design goals. Most have focused on its state-of-the-art development environment and rich class library. In this book, the author focuses instead on performance and/or standalone code generation.
- **Towards Object-centric Time-traveling Debuggers (Cat. 2):**
  Object-centric debugging aims to make debugging object-oriented programs easier by focusing debugging operations on specific objects. Time-shifting debuggers allow developers to explore executions in both directions of time. In this work, the authors present SeekerOC, a prototype time-shifting debugger that provides object-centric debugging support.
- **Using Moose platform for the implementation of a Software Product Line according to model-based Delta-Oriented Programming (Cat. 2):**
  Software product line engineering enables the reuse of common functionality to implement a set of products. In this work, the authors introduce a framework that supports model-based engineering to better manage the variability of product lines at a high level of abstraction. An industrial use case is proposed to illustrate the interest of the tools whose prototype is presented.
- **Transformation-based Refactorings: a First Analysis (Cat.3.):**
  Refactorings are well-known behavior preserving transformations. Little work exists on the analysis of their implementation and in particular how traditional refactorings might be composed from smaller, reusable, parts.In this work, authors focus on the possibilities to reuse transformations independently from the behavior preserving aspect of refactoring.
- **Can we automatically generate class comments in pharo? (Cat. 3, Best Paper):**
  Code comments support developers in understanding and maintaining codebases. In the

Pharo environment, code comments are the primary form of code documentation and typically convey information ranging from high-level design descriptions to low-level implementation details. In this work, the authors adopt the stereotype-based approach to automatically generate class comments in the pharo programming environment.

Acknowledgments

We would like to thank the Lam Research Corporation for their financial contribution to provide awards for the best papers.