2nd International Workshop on Advanced Software Development Tools and Techniques (WASDeTT): Tools for Software Maintenance, Visualization, and Reverse Engineering

Holger M. Kienle
University of Victoria, Canada
hkienle@acm.com

Leon Moonen
Simula Research Laboratory, Norway
leon.moonen@computer.org

Michael W. Godfrey
University of Waterloo, Canada
migod@uwaterloo.ca

Hausi A. Müller
University of Victoria, Canada
hausi@cs.uvic.ca

Abstract

The objective of the 2nd International Workshop on Advanced Software Development Tools and Techniques (WASDeTT) is to provide interested researchers with a forum to share their tool building experiences and to explore how tools can be built more effectively and efficiently. This workshop specifically focuses on tools for software maintenance and comprehension and addresses issues such as tool-building in an industrial context, component-based tool building, and tool building in teams.

1. Introduction and Rationale

The motivation for this workshop is the fact that tools and tool building play an important role in applied computer science research [5]. The tangible results of research projects are often embodied in a tool. Even though tool building is a popular technique to validate research (e.g., proof-of-concept prototyping followed by user studies), it is neither simple nor cheap to accomplish.

Given the importance of tool building and the significant cost associated with it, this workshop gathers interested researchers and to provide them with a forum to share their tool building experiences and to explore how tools can be built more effectively and efficiently. Thus, the workshop is not so much about the finished product—a tool's novel features and algorithms—but about *how* the tool was designed and built.

This workshop is the second installment of the *International Workshop on Advanced Software Development Tools and Techniques* (WASDeTT). We hope that WASDeTT will become a regular event, attracting researchers from software engineering and related areas. The first WASDeTT was held at the 22nd European Conference on Object-Oriented Programming (ECOOP 2008) in Cyprus.[1] This workshop focused on tools that are implemented in object-oriented languages and/or target object-oriented software. In contrast, this workshop specifically addresses tools that support maintenance and program comprehension of software systems. The aim of the workshop is to foster interactions between researchers that are constructing such tools, and to advance the state-of-the-art in tool building.

2. Workshop Topics

General topics of interest include questions such as: Should tool building remain a craft? Should researchers strive towards improving promising prototypes so that they can compete with the quality of professional offers—and should this effort be fostered and rewarded by academia? What are the positive lessons learned in building tools? What are the (recurring) pitfalls in tool building? What are the good practices and techniques? How to integrate and combine independently developed tools? Are there architectures and patterns for tool building in the research community? How to compare or benchmark tools?

For this workshop we are especially interested in experiences of speakers and participants that relate to the following topics:

Tool building in industry: There are examples of industrial tools that have started as academic prototypes. For example, the Bauhaus reverse engineering tool originated as part of a dissertation [8] and is now commercialized by Axivion (`www.axivion.com`). Other examples of academic spin-offs are Legasys Corporation in Canada, which

[1] `http://smallwiki.unibe.ch/wasdett2008/`

offered services in software maintenance automation for six years [2], and the Software Improvement Group in the Netherlands (`www.sig.nl`). The latter was founded by two Ph.D. students and now employs more than 30 people. In this context, we would like to discuss issues such as how to get the foot into industry's door for conducting case studies and user studies, and successful business models to transition a research tool into a commercial offering. Also, how to effectively ensure tool stability for commercial customers while continuing to use the tool as a testbed to try out now research ideas?

Component-based tool building: Researchers are increasingly leveraging components to assemble their tools instead of building them from scratch. Examples of components are off-the-shelf (OTS) products—commercial as well as open source—such as Eclipse [10], Rational Rose [4], Emacs [14], Visio [15], Graphviz [12], Source Navigator [11], and GCC [3]. While many researchers are leveraging OTS products, comparably few experiences and lessons learned are described in the literature (e.g., [13] [1] [6]). In this context, we would like to discuss concrete experiences and lessons learned of component-based building of software maintenance and comprehension tools. Question of interest are, for example: How to assess and select suitable OTS products? How to customize a certain OTS product (via its API or scripting)? How to interoperate with a certain OTS product?

Tool building in teams: Often tools are developed by a single researcher over a few years as part of his or her thesis or dissertation. These tools are typically prototypes that are abandoned after the degree is completed. In contrast, there are also tools that are developed and maintained over many years by a significant team of developers. Examples of such tools are Bauhaus (University of Bremen and University of Stuttgart) [9] and Rigi (University of Victoria) [7]. In this context, we would like to discuss how team size and team diversity impacts tool building, and how to manage larger teams. Especially, is there a need to introduce more formality in the tool-development process? And how can this be achieved without stifling creativity in research?

3. Workshop Activities and Tangible Outcomes

The workshop features presentations by invited speakers that will talk about their tool building experiences in an academic and industrial context. The purpose of the presentations is to convey lessons learned and to point out open issues that can generate interesting discussion. We plan to have an interactive workshop with ample time for discussion and a break-out session. After the workshop, we would like to solicit feedback from each participant about the most important insight that he or she has learned in the workshop.

In order to preserve and disseminate results, talks, position papers and other materials that the workshop generates will be published on the workshop's web site at

`http://wasdett2.wikispaces.com/`.

Furthermore, the results from this workshop will be also summarized in the next WASDeTT.

As concrete outcome, we aim to distill practical results and experiences so that other researchers can apply them to improve upon their own tools and upon the way that they build them. As a result, we expect that this workshop will contribute towards advancing the current state-of-the-art in tool building.

References

[1] D. Coppit and K. J. Sullivan. Multiple mass-market applications as components. *22nd ACM/IEEE International Conference on Software Engineering (ICSE'00)*, pages 273–282, June 2000.

[2] J. R. Cordy. Comprehending reality—practical barriers to industrial adoption of software maintenance automation. *11th IEEE International Workshop on Program Comprehension (IWPC'03)*, pages 196–206, May 2003.

[3] T. R. Dean, A. J. Malton, and R. Holt. Union schemas as a basis for a C++ extractor. *8th IEEE Working Conference on Reverse Engineering (WCRE'01)*, pages 59–67, Oct. 2001.

[4] A. Egyed and P. B. Kruchten. Rose/Architect: A tool to visualize architecture. *32rd IEEE Hawaii International Conference on System Sciences (HICSS'99)*, Jan. 1999.

[5] R. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8):491–506, June 2002.

[6] H. M. Kienle. Building reverse engineering tools with software components: Ten lessons learned. *14th IEEE Working Conference on Reverse Engineering (WCRE 2007)*, pages 289–292, Oct. 2007.

[7] H. M. Kienle and H. A. Müller. The Rigi reverse engineering environment. *1nd International Workshop on Advanced Software Development Tools and Techniques (WASDeTT 1)*, July 2008. To appear.

[8] R. Koschke. *Atomic Architectural Component Recovery for Program Understanding and Evolution*. PhD thesis, University of Stuttgart, Germany, 2000.

[9] R. Koschke. Zehn Jahre WSR – Zwölf Jahre Bauhaus. *10th Workshop Software Reengineering (WSR 2008)*, May 2008. `http://www.informatik.uni-bremen.de/st/papers/bauhaus-wsr08.pdf`.

[10] R. Lintern, J. Michaud, M. Storey, and X. Wu. Plugging-in visualization: experiences integrating a visualization tool with Eclipse. *ACM Symposium on Software Visualization (SoftVis'03)*, pages 47–56, June 2003.

[11] D. L. Moise and K. Wong. An industrial experience in reverse engineering. *10th IEEE Working Conference on Reverse Engineering (WCRE'03)*, pages 275–284, Nov. 2003.

[12] G. C. Murphy, D. Notkin, and K. J. Sullivan. Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27(4):364–380, Apr. 2001.

[13] S. P. Reiss. Program editing in a software development environment (draft). `http://www.cs.brown.edu/~spr/research/desert/fredpaper.pdf`, 1995.

[14] P. Tonella, G. Antoniol, R. Fiutem, and E. Merlo. Points-to analysis for program understanding. *5th IEEE International Workshop on Program Comprehension (IWPC'97)*, pages 90–99, Mar. 1997.

[15] Q. Zhu, Y. Chen, P. Kaminski, A. Weber, H. Kienle, and H. A. Müller. Leveraging Visio for adoption-centric reverse engineering tools. *10th IEEE Working Conference on Reverse Engineering (WCRE'03)*, pages 270–274, Nov. 2003.