Science of Computer Programming 79 (2014) 1–5

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Guest editors' introduction to the 4th issue of Experimental Software and Toolkits (EST-4)

Kim Mens [a], M.G.J. van den Brand [b,*], Holger M. Kienle [c]

[a] Université catholique de Louvain, Place Sainte Barbe 2, B-1348 Louvain-la-Neuve, Belgique
[b] Eindhoven University of Technology, Den Dolech 2, NL-5612 AZ Eindhoven, The Netherlands
[c] Freier Informatiker, Germany

ARTICLE INFO

Article history:
Received 5 December 2012
Accepted 5 December 2012
Available online 27 December 2012

Keywords:
Experimental software
Tools
Tool building
Software development
Open science

ABSTRACT

Experimental software and toolkits play a crucial role in computer science. Elsevier's Science of Computer Programming special issues on Experimental Software and Toolkits (EST) provide a means for academic tool builders to get more visibility and credit for their work, by publishing a paper along with the corresponding system in a peer-reviewed journal. Typically, tools are presented from both a user and a developer perspective, addressing tool-building issues such as architecture and design, requirements, methodologies and process aspects. This is already the fourth edition of EST with no less than 17 published systems covering application areas ranging from software analysis and visualization to teaching and software development support.

© 2012 Elsevier B.V. All rights reserved.

Background and motivation

In computer science – and software engineering research in particular – experimental systems and toolkits play a crucial role. Some research is about novel tools or systems that support software engineers, researchers, teachers or other stakeholders directly in their various activities. Some academic tools are developed to help validate certain research ideas. Others are developed as, or have evolved into, frameworks or tool suites that can be used as a backbone for a whole range of other tools, or even into commercial products. Yet other systems are just early prototypes of novel research ideas. Academic tools thus come in various shapes and flavors, but it cannot be denied that they have become an essential part of the (software) research landscape.

Many research tools are left abandoned at the early prototype stage. This is expected due to the nature of research. An early prototype may be used as a vehicle to support or refute a research hypothesis, or to demonstrate the feasibility of a novel idea. Once the prototype has served its purpose and the results have been published, there may be no further need or desire to evolve it any further. Some tools, however, do evolve further and may even develop an ecosystem of users and contributors, serving the role of a stable backbone for a research community. Not surprisingly, maintaining such a tool is a major task.

Brooks already pointed out that effort in software does not stop at the basic functionality that it delivers [4,10]. Whereas a tool prototype is just a program, written for one user, its creator, and is not expected to survive for a very long time, moving from a tool prototype to a *tool product* easily takes three times the effort. This effort is needed to make it truly usable for a broader range of tool users and entails, for instance, enhanced error checking/messages, platform independence, bug fixing, testing, robustness, and generalization, as well as high-quality documentation and a website that allows user–developer

* Corresponding author.
E-mail addresses: kim.mens@uclouvain.be (K. Mens), m.g.j.v.d.brand@tue.nl (M.G.J. van den Brand), hkienle@acm.org (H.M. Kienle).

0167-6423/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2012.12.005

interactions (e.g., involving a Wiki, a blog, and bug-tracking). Similarly, moving from a tool prototype to a *tool platform*, which can take on the form of a suite, library, framework or domain-specific language, may also take three times the effort of developing a simple prototype. *Combining* tool product and tool platform multiplies both efforts. Thus, to move a tool prototype to its most advanced stage it is estimated to take about nine times the effort. Whereas these numbers are meant as rough indicators only, they show that a significant effort is needed to advance a research prototype into a more mature state.

Unfortunately, it is often hard for academic tool developers to get sufficient credit or resources for their tool-building efforts. Funding agencies, academic authorities, evaluation agencies and even fellow software researchers are not sufficiently aware – or are not easily convinced – of the importance of tool building in research. Publication and citation counts have become almost the sole criteria for research quality—little else seems to matter anymore, and as a result other valid considerations are being neglected. This indifferent or sometimes negative attitude towards tool building in computer science research has a detrimental effect on tool-oriented research. Since publications are all that count, why would researchers bother in creating, maintaining, or evolving their research tools, if they can barely get credit for that work?

EST and related efforts

The WASDeTT workshop series (http://wasdett.wikispaces.com/) and Elsevier's *Science of Computer Programming* special issues on *Experimental Software and Toolkits* (EST) are our modest contributions to address this problem and provide a means to tool builders to get at the same time credit for their work and more visibility for their tool, by providing a venue to publish both the tool and the corresponding system in a peer-reviewed journal.

Besides EST, there are a number of efforts in the software engineering research community that address issues surrounding tool building. There has been a special issue on *Tool Building in Formal Methods* published at Wiley's *Journal of Software: Practice and Experience*. The editors of the special issue address concerns similar to ours when they point out that "building tools represents a big amount of work, which is not greatly recognized, especially regarding to research evaluation criteria" [9]. The *12th International Working Conference on Source Code Analysis and Manipulation* (SCAM 2012) has a *Tool Papers* track that asks for submissions that address the user perspective (description of the tool's "most appealing use cases") as well as the developer perspective (description of the tool's "architecture and …its inner workings" so that it "can benefit others engaged in designing and building tools"). The *16th European Conference on Software Maintenance and Reengineering* (CSMR 2012) has a *Tool Demonstration* track that explicitly asks papers to address the tool's underlying infrastructure and challenges to its adoption [13].

WASDeTT and EST allow a more elaborate discussion of, for instance, the underlying infrastructure/architecture, identified implementation issues, and validation of the tool. Furthermore, at conferences, like SCAM and CSMR, tool papers are actually short demos and the reviewers and/or audience cannot really use the tool. EST requires the reviewers to install, use and evaluate the tools as well.

From a broader perspective, published tools enable more open and collaborative forms of research. In academia there is a growing recognition that scientific data and artifacts should be made freely available to speed up scientific advances (so-called open science and open science data). Tools that are freely available to researchers enable, for instance, more meaningful tool comparisons (e.g., scalability benchmarks) and comparative user studies (e.g., effectiveness of different visualization or interaction techniques). Freely available tool platforms make it easier for researchers to build new research prototypes by relying on those platforms. Open repositories, like SourceForge, allow an easy distribution of the source code of (academic/experimental) tools in order to encourage both the use and further development of the tools. From this perspective, SourceForge and the like can be seen as infrastructure that enable open science. Sadly, however, distributing a tool via such infrastructure does not give its authors academic credit.

EST is our contribution to open science by encouraging authors to package their tools in such a way that they are readily accessible to and reusable by fellow researchers. The EST reviewers serve as gatekeepers by installing and trying out the submitted tools. Another example of a step towards open science in the community is apparent by the *20th International Symposium on the Foundations of Software Engineering* (FSE-20), which in its research track encourages the submission of research artifacts to "facilitate reproducibility and follow-up research". Tools are one example of research artifact and "making tools publicly available" is explicitly encouraged.

This special issue

This fourth edition of the EST special issue was announced at the *3rd International Workshop on Academic Software Development Tools and Techniques* (WASDeTT-3). WASDeTT-3 was co-located with ASE'10 and held on September 20, 2010 in Antwerp, Belgium. It was organized by Anthony Cleve and us. WASDeTT-3 received 14 submissions of which 11 were accepted for presentation and inclusion into the informal workshop proceedings [11]. WASDeTT-3 also featured a stimulating invited talk from Paul Klint on *Building Academic Software Tools: Do's and Don'ts*.[1]

While authors of accepted WASDeTT-3 papers were encouraged to submit to the EST special issue, we decided to open the call (http://www.win.tue.nl/~mvdbrand/SCP-EST/) to attract tools beyond the scope that ASE usually attracts. In response,

[1] http://www.info.fundp.ac.be/wasdett2010/wp-content/uploads/2010/10/Klint-slides-WASDeTT3.pdf.

we received an exceptionally high amount of high-quality submissions. No less than 28 tools or systems – each with an accompanying paper – were submitted, of which eventually 17 got accepted after two or more thorough reviewing rounds.

Given the large amount of submissions, the review process proved to be an arduous task. In particular, given the significant effort for a single reviewer to evaluate both a tool or system and the different revisions of the associated paper, with a few notable exceptions no reviewer was assigned more than one paper. To ensure quality reviews worthy of this journal, for the review process every single paper was assigned to three or four reviewers, as well as one of the special issue editors for follow-up during the reviewing process, to ensure consistency among how all papers were handled. We were thus faced with the humongous task of finding around 90 different reviewers, for which we contacted a few hundred established researchers. This, plus the inevitable delays in reviews and revisions, lead to the rather long time interval between initial submission and final publication of the printed journal issue. Fortunately, as soon as papers successfully made it through the reviewing process, they were published electronically shortly afterwards by Elsevier as *Article in Press*, so that readers could start accessing it and authors could start referring to it. We are therefore very grateful of the help we received from Elsevier (and Bas van Vlijmen in particular) throughout the reviewing process. We are equally grateful to all reviewers who took the time and effort to not only review a paper, but also to install, try out and provide detailed feedback on the accompanying system. Finally, we thank the authors for their high quality submissions and for having had the patience and energy to undergo this lengthy process—but in the end we are confident it was all worth the while!

This special issue contains 17 papers, discussing 17 software engineering "tools" in the broader sense:

AspectMaps [7]	http://pleiad.cl/aspectmaps
CoreASM [8]	http://www.coreasm.org/
DiaSuite [3]	http://diasuite.inria.fr/
Disnix [25]	http://nixos.org/disnix
FeatureIDE [21]	http://www.fosd.de/featureide
Hapao [2]	http://objectprofile.com/pier/Products/Hapao
JACCIE [14]	http://inf2.w3.rz.unibw-muenchen.de/Tools/Syntax/english/index.html
JBlnsTrace [5]	http://www.loria.fr/~casertap/jbinstrace.html
jCOLIBRI2 [17]	http://www.jcolibri.net
JP2 [19]	http://jp-profiler.origo.ethz.ch
NiCad [6]	http://www.txl.ca/nicaddownload.html
Process Enactment [15]	http://pet.codeplex.com/
Salespoint [27]	http://www.salespoint-framework.org/
Softwarenaut [16]	http://scg.unibe.ch/softwarenaut
Solid* [18]	http://www.solidsourceit.com/
Sourcerer [1]	http://www.sourcerer.ics.uci.edu/
Taupe [20]	http://www.ptidej.net/research/taupe/

All of the tools are available for download, and nearly all developers have told us that their tool will be further actively developed and/or maintained.

Readers of an EST article can typically expect that a tool is presented from both *a user and a developer perspective*. On the one hand, the article describes the tool's research contributions and presents examples and case studies that illustrate how the tool can be applied. On the other hand, the article looks "under the hood" of the tool, addressing issues such as the tool's architecture and design, tool requirements (e.g., scalability) and how they have been addressed at the technical level, methodologies and process aspects for tool development, and managerial issues to ensure a tool's long-term survival. We believe that especially this development perspective has been neglected in the research community and that more research and publications should explore this perspective in more depth. In this vein, we are pleased that this special issue contains a paper about the NiCad tool experience, which nearly exclusively addresses the development perspective. This article serves as an example of a unique kind of research article that explores tool-building in a systematic and in-depth manner. Such articles deserve equal recognition in the research community compared to more established kinds of research articles.

Table 1 provides an overview of the tools' domains, execution platforms, employed licenses, tool types, and level of maturity. The tools cover a broad range of application areas, technologies and techniques, ranging from software visualization and static and dynamic code analysis to teaching and software development support. It is convenient that almost all of the tools are not bound to a specific operating system, which is typically achieved by building on top of a cross-platform infrastructure, most prominently Java. The most popular licenses are GNU GPL/LGPL (6 tools) and MIT (3 tools). Not all tools have a license, but the authors can be contacted for details in such cases. Solid* is the only tool that is available commercially, but it is free for research usage. We asked the authors of each tool which of the following characteristic(s) best describe it: stand-alone tool, tool suite, plug-in, and framework. Their responses show that we received a healthy mix of different tool types, and that several tools even have multiple types to give users and developers more freedom in how to integrate the tool into their environments. JBlnsTrace is a bit of an outlier because it is a Java agent deployed as a JAR file. We also asked the authors to judge the maturity of their tools according to the following categories: early prototype, advanced prototype, mature tool, or industrial-strength. All tools are beyond the early prototype stage, with Hapao and Solid* being the most mature ones.

Table 1
Overview of the tools.

Tool	Purpose	Platform/OS	License	Type	Maturity
AspectMaps	Visualization of aspect-oriented programs	Mac OS, Linux, Windows	MIT	Stand-alone	Advanced prototype
CoreASM	Language and tool environment for abstract state machines	Windows, Linux, Mac OS	Academic Free License 3.0	Framework, suite, or stand-alone	Advanced prototype, almost mature
DiaSuite	Building Sense/Compute/Control applications	Cross-platform	INRIA-specific	Suite	Mature
Disnix	Deploying service-oriented systems	Cross-platform	LGPL	Suite	Advanced prototype
FeatureIDE	Feature-oriented software development	Java	GPL	Framework	Advanced prototype
Hapao	Visualization of test coverage	Smalltalk: Pharo, VisualWorks	MIT	Suite	Industrial-strength
JACCIE	Teaching compiler generation techniques	JDK (1.6.31)	Free for teaching	Suite	Advanced prototype, almost mature
JBlnsTrace	Instrumenting and tracing Java bytecode	VM with Java agent service	Proprietary	Java agent	Advanced prototype
jCOLIBRI2	Case-Based Reasoning applications	Java	LGPL	Framework	Mature
JP2	Collecting dynamic bytecode metrics for Java	Cross-platform	GPL	Stand-alone	Advanced prototype
NiCad	Clone detection in source code	Linux, OS X	Modified BSD attribution	Stand-alone	Mature
Process enactment	Software process tool support	Windows	Apache 2.0	Stand-alone and plug-in	Mature
Salespoint	Teaching object-oriented software development	Java (>= 5.0)	Free	Framework	Mature
Softwarenaut	Architecture discovery	OS X, Linux, Windows	MIT	Stand-alone	Advanced prototype
Solid*	Visual exploration of source code	Windows (.NET 4.0)	free for research	Suite	Industrial-strength
Sourcerer	Analysis of open source code	Java/Eclipse	GPL	Suite	Partly advanced prototype, partly mature
Taupe	Visualization and analysis of data recorded by eye-tracking devices	Java	GPL	Stand-alone and plug-in	Mature

Looking back, looking forward

We have now pursued the topic of tool-building over several years and as a result there are a number of resources related to EST and the WASDeTT series that may be of interest to readers of this special issue.

We are pleased that EST is now in its fourth iteration and that it is supported by many authors and reviewers that have greatly contributed to its success. The previous EST special issues have been published as the following SCICO issues:

- EST-3: Volume 75, Issue 4, April 2010 (associated with WASDeTT-1) [24]
- EST-2: Volume 72, Issues 1–2, June 2008 [23]
- EST-1: Volume 69, Issues 1–3, December 2007 [22]

In addition to EST, we felt the need also to have a less formal but more interactive platform for interested researchers to exchange ideas that relate to tool-building. This need is addressed by the WASDeTT workshop series, whose foundation was laid at the *10th Workshop on Object-Oriented Reengineering* (WOOR 2007) at ECOOP 2007, and the first WASDeTT was held a year later at ECOOP 2008. The results of WASDeTT-1 are summarized in a detailed workshop report [26], and our observations about current and emerging tool-building trends have been described in an IEEE Software column [12]. While WASDeTT-2 (http://wasdett2.wikispaces.com/) featured talks and emphasized discussions, WASDeTT-3 has published informal proceedings of accepted papers [11].

Looking ahead, we are seeing that the software engineering community is becoming more mature and that more promising initiatives and events, like the ones mentioned above, are being introduced. We call on interested parties to further strengthen initiatives geared towards tool-building that enable open science, reproducibility and academic recognition/credits. It would be desirable, for instance, to have a dedicated platform for publishing tools, workflows and data, and to build up a collaborative knowledge base that collects tool-building experiences in the form of patterns, architectures, processes, etc. We are interested to hear from readers of this special issue on how to advance EST and its goals.

The call for a fifth special issue has been launched (http://www.win.tue.nl/~mvdbrand/SCP-EST/). This call is related to the ACademics Modeling with Eclipse (ACME) workshop (http://www.acme-workshop.org/). The focus in this call is on

the development of modeling tools. Furthermore, if the opportunity arises a new edition of WASDeTT will be organized in combination with another EST special issue.

References

[1] Sushil Bajracharya, Joel Ossher, Cristina Lopes, Sourcerer: an infrastructure for large-scale collection and analysis of open-source code, Science of Computer Programming 79 (2014) 241–259.
[2] Alexandre Bergel, Vanessa Peña, Increasing test coverage with Hapao, Science of Computer Programming 79 (2014) 86–100.
[3] Benjamin Bertran, Julien Bruneau, Damien Cassou, Nicolas Loriant, Emilie Balland, Charles Consel, Diasuite: a tool suite to develop Sense/Compute/Control applications, Science of Computer Programming 79 (2014) 39–51.
[4] Frederick P. Brooks Jr., The Mythical Man–Month (anniversary ed.), Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, 1995.
[5] Pierre Caserta, Olivier Zendra, JBInsTrace: a tracer of Java and JRE classes at basic-block granularity by dynamically instrumenting bytecode, Science of Computer Programming 79 (2014) 116–125.
[6] James R. Cordy, Chanchal K. Roy, Tuning research tools for scalability and performance: the NiCad experience, Science of Computer Programming 79 (2014) 158–171.
[7] Johan Fabry, Andy Kellens, Simon Denier, Stéphane Ducasse, AspectMaps: extending Moose to visualize AOP software, Science of Computer Programming 79 (2014) 6–22.
[8] Roozbeh Farahbod, Vincenzo Gervasi, Uwe Glässer, Executable formal specifications of complex distributed systems with CoreASM, Science of Computer Programming 79 (2014) 23–38.
[9] Frédéric Gervais, Benoît Fraikin, Tool building in formal methods, Software: Practice and Experience 41 (2) (2011) 131–132.
[10] Poul-Henning Kamp, The hyperdimensional tar pit, Communications of the ACM 55 (3) (2012) 52–53.
[11] Holger M. Kienle (Ed.) Third International Workshop on Academic Software Development Tools and Techniques, September 2010. http://www.info.fundp.ac.be/wasdett2010/wp-content/uploads/2010/08/WASDeTT-3.pdf.
[12] Holger M. Kienle, Adrian Kuhn, Mim Mens, Mark van den Brand, Roel Wuyts, Tool building on the shoulders of others, IEEE Software 26 (1) (2009) 22–23.
[13] Holger M. Kienle, Mircea Lungu, Welcome from the tool demonstration chairs, in: Proceedings of the 2012 16th European Conference on Software Maintenance and Reengineering, CSMR'12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 519–520.
[14] Nico Krebs, Lothar Schmitz, Jaccie: a Java-based compiler–compiler for generating, visualizing and debugging compiler components, Science of Computer Programming 79 (2014) 101–115.
[15] Marco Kuhrmann, Georg Kalus, Manuel Then, The process enactment tool framework—transformation of software process models to prepare enactment, Science of Computer Programming 79 (2014) 172–188.
[16] Mircea Lungu, Michele Lanza, Oscar Nierstrasz, Evolutionary and collaborative software architecture recovery with Softwarenaut, Science of Computer Programming 79 (2014) 204–223.
[17] Juan A. Recio-Garca, Pedro A. González-Calero, Belén Déaz-Agudo, jcolibri2: a framework for building case-based reasoning systems, Science of Computer Programming 79 (2014) 126–145.
[18] Dennie Reniers, Lucian Voinea, Ozan Ersoy, Alexandru Telea, The Solid* toolset for software visual analytics of program structure and metrics comprehension: from research prototype to product, Science of Computer Programming 79 (2014) 224–240.
[19] Aibek Sarimbekov, Andreas Sewe, Walter Binder, Philippe Moret, Mira Mezini, JP2: Call-site aware calling context profiling for the Java Virtual Machine, Science of Computer Programming 79 (2014) 146–157.
[20] Benoît De Smet, Lorent Lempereur, Zohreh Sharafi, Yann-Gaël Guéhéneuc, Giuliano Antoniol, Naji Habra, Taupe: visualizing and analyzing eye-tracking data, Science of Computer Programming 79 (2014) 260–278.
[21] Thomas Thüm, Christian Kästner, Fabian Benduhn, Jens Meinicke, Gunter Saake, Thomas Leich, FeatureIDE: an extensible framework for feature-oriented software development, Science of Computer Programming 79 (2014) 70–85.
[22] M.G.J. van den Brand, Guest editor's introduction: experimental software and toolkits (est), Science of Computer Programming 69 (1–3) (2007) 1–2.
[23] M.G.J. van den Brand, Guest editor's introduction: Second issue of experimental software and toolkits (est), Science of Computer Programming 72 (1–2) (2008) 1–2.
[24] Mark G.J. van den Brand, Kim Mens, Editorial: Guest editors' introduction to the 3rd issue of experimental software and toolkits (est): a special issue on academic software development tools and techniques (wasdett 2008), Science of Computer Programming 75 (4) (2010) 214–215.
[25] Sander van der Burg, Eelco Dolstra, Disnix: a toolset for distributed deployment, Science of Computer Programming 79 (2014) 52–69.
[26] Roel Wuyts, Holger M. Kienle, Kim Mens, Mark van den Brand, Adrain Kuhn, Academic software development tools and techniques: report on the 1st workshop WASDeTT at ECOOP 2008, in: Patrick Eugster (Ed.), Object-Oriented Technology. ECOOP 2008 Workshop Reader, in: Lecture Notes in Computer Science, vol. 5475, Springer Verlag, 2009, pp. 87–103.
[27] Steffen Zschaler, Birgit Demuth, Lothar Schmitz, Salespoint: a Java framework for teaching object-oriented software development, Science of Computer Programming 79 (2014) 189–203.