Guest editor's introduction: Second issue of experimental software and toolkits (EST)

1. Goal and scope

Academic research in software and software engineering may involve the development of huge software systems in order to perform all kinds of experiments. This can be compared to experiments in physics or chemistry. In these areas it is not uncommon to build huge installations, for instance a particle accelerator, before any experiments on particles can be performed. The design and implementation of such a particle accelerator is, in fact, already research, it may be necessary to develop new techniques in order to build such an installation. This is comparable with the situation in modern computer science research, the design and implementation of software tools to perform experiments has become research as well. The focus of researchers developing such experimental software systems is mainly on the software development itself. The software developer has some idea, algorithmic or functional, and in order to prove this idea software has to be designed, implemented or adapted.

Some of these systems are distributed to fellow researchers in order to perform other experiments and may become very popular in the (research) community. This way of knowledge sharing is done continuously and is increasing. The open source community provides a mechanism to distribute software in order to allow others to do experiments. This is exactly what almost every software developer wants, build software that is used by others.

Another issue of academic software development is related to reproducibility. Physics or chemistry papers about experiments contain a lot of details in order to enable other researchers to replay the experiments and so to validate the results described in the papers. This is common practice and sometimes reveals that the authors misinterpreted the results, for instance with respect to cold fusion. Papers describing the implementation aspects of software tools are in general hard to read and reproducing the results is in fact often impossible, because it would involve a complete reimplemention and the effort is too big. Some conferences offer tool builders the possibility to demonstrate their developed tools. This often results in a short conference publication. However, the tool itself can not be tested or used. If the software is not distributed via the open source community, there is no accessible way to obtain the software.

Distributing the software via the open source community does not yield any academic credits. The academic credits can only be obtained via journal or conference publications. However, software development is almost always in conflict with papers that must be written, unless the research group is big enough to employ engineers that do the coding and maintenance of the developed systems. If the user community grows and the requests for enhancements and improvements increases, the balance between software development and writing scientific papers can tip the wrong way. These special issues of Science of Computer Programming are a way to create a forum where software engineers can publish their systems together with a (short) paper to ensure the academic recognition they deserve.

This is the second special issue on Experimental Software and Toolkits. We have invited authors to submit software systems together with a (short) paper. We asked the referees not only to referee the paper, but to actually install and use the software system. The referees had, with respect to the systems, to look at various aspects:

- ease of installation,
- quality of (user) documentation,
- ease of usage, and
- applicability to the intended domain.

EST offers a forum to publish systems and to get the academic credits, but it also offers a way of archiving the developed systems for the future. An improvement of conference and workshop tool demonstrations is that a broader audience is able to download and use the systems. The main difference with the open source community is that independent referees from the academic community gave a verdict on the system and the paper.

0167-6423/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
2. Submissions

The call for the second special issue was not restricted to a specific application domain. Four systems were submitted and eventually three of these systems were accepted. This special issue contains, however, five systems, the reason for this is that there were a few authors of submissions from the first issue who asked for more time to improve either the paper and/or the system. They are published in this issue.

The quality of the submitted systems and corresponding papers was very high, given the fact that we only rejected one system. However, the time between submission and actual publication is far too long. It is still very hard to find (qualified) referees. Nevertheless, thanks to pressure from, amongst others, some of the authors, this second issue is now a fact. The systems cover a broad range of applications from pretty-printing to program transformations.

3. Academic software development

In the introduction of the first issue I wrote that I hoped that EST would lead to a forum where academic software engineers can publish their systems and get scientific credits for their work. I believe this is happening. Academic software engineers start sending e-mails when a next call for systems is published. Together with Kim Mens of the University of Louvain (Belgium) we decided to launch a workshop dedicated to academic software development, the International Workshop on Advanced Software Development Tools and Techniques (WASDeTT’2008). The goal of this workshop is to enable academic software engineers to demonstrate their tooling and to discuss the state-of-art with respect to academic software development. The tool demonstrators will be invited to submit their tools to the third special issue of EST.

I observe that academic software development is getting more mature. Systems that start as a prototype to prove that some idea could work are redesigned and reimplemented to become more mature and to fulfill the requirements posed by fellow researchers. Of course, we are not yet in the phase where we first formulate our requirements, because when building this initial prototype the only goal is to make it work and prove that your ideas are correct. Upon success, software engineers realise that the way of working has to improve and be made more efficient. This is also the phase where tool builders decide to improve the user documentation, start writing the architectural design documents and set up facilities to install and distribute their software in a more accessible way. At this point, such tooling is mature enough to be submitted to EST in order to obtain the academic credits and to archive it for the future.

M.G.J. van den Brand
Eindhoven University of Technology,
Department of Mathematics and Computer Science,
Den Dolech 2,
NL-5612 AZ Eindhoven,
The Netherlands
E-mail address: m.g.j.v.d.brand@tue.nl.

2 April 2008
Available online 25 April 2008