Editor: **Diomidis Spinellis**
Athens University of Economics and Business, dds@aueb.gr

Software Tools Research: SPLASH Panel Discussion

Dennis Mancl and Steven Fraser

AT THE RECENT SPLASH (Systems, Programming, Languages and Applications: Software for Humanity) conference, one of us (Steven Fraser) organized an international group of experts to discuss challenges in software tools research.[1] The panelists included Kendra Cooper (University of Texas, Dallas), Jim "Cope" Coplien (Gertrud & Cope), Junilu Lacar (Cisco Systems), Ruth Lennon (Letterkenny Institute of Technology), Diomidis Spinellis (Athens University of Economics and Business), and Giancarlo Succi (Free University of Bolzano-Bozen).

The discussion interwove three threads—tool use, development, and education—and the panelists took a critical look at how well tools serve the needs of software professionals, managers, and academics. Their passion for the topic was reflected through some heated exchanges, even during the opening statements.

Tool Use and Misuse

Cooper contrasted the perspective of tool users in industry who are building and integrating large production systems with the researchers who build small prototype tools to explore software engineering ideas.

Lacar spoke for industry: he wants tools research to focus more on creating tools and languages that are clean, clear, and simple. Tools must be focused on the job tasks of software professionals; they must help individual developers think and support clear communication within development teams.

Succi described research that collects data about how tools are really used. Most software developers only use eight to 12 tools, even when more tools are available. Succi believes that tool users should reflect on their tool usage and workflows, and tools should report on how they are used. This would allow developers to assess the real "business value" of their tools and give them input on how to adjust the tools and their development processes to produce value more effectively.

Coplien complained about the misuse of tools: "The way managers manage people is through adherence to the tools. So it isn't that the tools guide the way, they enforce the managers' will to implement workflows conceived 10 years ahead of the time that they are used, that are 10 years out of date with prevailing knowledge and maturity. Tools are the enforcer of the management prisons, they're the guards at the gates of the prisons, to keep the developers from being innovative."

Spinellis concurred, lamenting the fact that his UMLGraph tool, which allows designers to create simple UML diagrams from a simple text-based specification language, was hijacked by its user community to generate UML diagrams directly from Java code.

From there, the conversation heated up. Coplien criticized the software development community for relying on tools, especially tools to support the design process. Commodity tools are fine, such as simple program editors and compilers for building software: they're the equivalent of hammers and saws for building houses. But when inexperienced software designers use design tools based on UML, they can easily confuse a pretty picture with a good quality design. When managers mandate agile management tools such as RallyDev, they might use the

0740-7459/13/$31.00 © 2013 IEEE

tool to enforce a non-agile waterfall development process. Coplien went on to declare that tools are evil because they encourage misuse: software professionals doing their jobs without thinking, learning, and communicating with others. "We don't need more research on tools, we need to get rid of the tools," he stated.

Lennon immediately pointed out the inconsistency of Coplien's complaints about tool use: "While you're giving out about tools, you're looking at your laptop and using a tool."

Succi also challenged Coplien's statement that tools are always bad by pointing out that the existing research literature on how users really use their tools is very sparse. Coplien brought up some of his work on using games to get team members to communicate and collaborate more effectively—for example, the ScrumKnowsy tool, a game designed to encourage more collaboration within a Scrum team.

Education

Lennon spoke for the learning community: the students at universities and technical institutes who use tools. The community of learners is large and diverse. Younger learners are already comfortable with using tools such as BitTorrent, YouTube, and Dropbox on a daily basis. More institutions are using virtual learning environments to allow learners to more conveniently access study materials. Some tools support hands-on learning experiences, whereas others permit students to view or listen to short 10-minute learning nuggets on the learner's own schedule.

Succi confessed his preference to do serious software development with minimal tools, while teaching development with more sophisticated tools: "When I have anything serious to do, I use vi (Unix text editor). However, when I have to train a young person in class, I use Eclipse."

Questions from the floor raised more issues about learning and communication. There is a lot of interest in simple Web-based learning tools, following the lead of Minute Physics and Khan Academy. Cooper discussed the "gamification" of applications: "I've been working on games for a while, serious educational games, and I see that they're trying to engage people and get people more involved with what they're working with."

Tool Development

Spinellis shared some of his experiences as a developer of open source software development tools. The increasing complexity of software systems, programming languages, and development environments is making the development of new tools more difficult. On the other hand, development and delivery environments like github support the sharing of open source tools and incorporate incremental improvements made by tool users. The best tools aren't big monoliths, they are small bricks that others can use to build on.

David Ungar, speaking from the floor, suggested that most software tools violate the object-oriented design principle that behavior should be combined with data. The Self language, developed by Ungar and Randall B. Smith at Xerox PARC in the late 1980s, used a different philosophy: instead of having an inspector to look at an object, Self just had the object appear on the screen as an outliner. So functionality that would normally be in a separate tool was part of the environment: "Every bolt had a handle on it, instead of needing a wrench." Ungar's approach can't replace all tools, but during the discussion, Spinellis pointed out that it is good to build more tools that do one thing well and can be composed with other tools.

Lacar addressed a question about "forking," where multiple versions of an open source tool are created in parallel. Forking is mostly good because it leads to competition and improvement, but it also causes a lot of confusion. When there is a fork in a popular tool, such as the Hudson/Jenkins continuous integration build management systems, users are left wondering, "What do we do now? Which way do we go?"

Lennon's final comment summarized the view of the panel's academics. To do exploratory work on tools, researchers need to go beyond a handful of small case studies, and companies need to be more open to sharing information: "If you don't trust, we can't get in and do the research. If you don't trust us to keep your data confidential, we'll never get there."

A complete transcript of the panel discussion can be found on this column's blog at www.spinellis.gr/tools.

Reference

1. S. Fraser et al., "Software Tools Research: A Matter of Scale and Scope—or Commoditization?" *Proc. Conf. Systems, Programming, and Applications: Software for Humanity* (SPLASH 12), G.T. Leavens, ed., ACM, 2012, pp. 59–62; doi:10.1145/2384716.2384740.

DENNIS MANCL is a Distinguished Member of Technical Staff at Alcatel-Lucent in Murray Hill, New Jersey. His current work is in agile software development and legacy software. Mancl received a PhD in computer science from the University of Illinois and is a senior member of IEEE and a member of ACM. Contact him at dennis.mancl@alcatel-lucent.com.

STEVEN FRASER is the director of the Cisco Research Center in San Jose and was previously a member of Qualcomm's Learning Center in San Diego. He received a PhD in electrical engineering from McGill University and is a senior member of both IEEE and ACM. Contact him at sdfraser@ieee.org.

See www.computer.org/software-multimedia for multimedia content related to this article.