How and When to Transfer Software Engineering Research via Extensions

David Shepherd*, Kostadin Damevski†, Lori Pollock‡
*ABB Corporate Research
940 Main Campus Dr., Raleigh, NC 27606
Email: david.shepherd@us.abb.com
†Virginia State University, Petersburg, VA
Email: damevski@acm.org
‡University of Delaware, Newark, DE
Email: pollock@cis.udel.edu

***Abstract*—It is often reported that there is a large gap between software engineering research and practice, with little transfer from research to practice. While this is true in general, one transfer technique is increasingly breaking down this barrier: extensions to integrated development environments (IDEs). With the proliferation of app stores for IDEs and increasing transfer effort from researchers several research-based extensions have seen significant adoption. In this talk we'll discuss our experience transferring code search research, which currently is in the top 5% of Visual Studio extensions with over 9,000 downloads, as well as other research techniques transferred via extensions such as NCrunch, FindBugs, Code Recommenders, Mylyn, and Instasearch. We'll use the lessons learned from our transfer experience to provide case study evidence as to best practices for successful transfer, supplementing it with the quantitative evidence offered by app store and usage data across the broader set of extensions. The goal of this 30 minute talk is to provide researchers with a realistic view on which research techniques can be transferred to practice as well as concrete steps to execute such a transfer.**

***Index Terms*—extensions, tech transfer, integrated development environment**

I. The Battlefield: Why Extensions Are An Effective Transfer Vehicle

While integrated development environments (IDEs) have existed for years, the barrier to IDE customization was, until recently, relatively high. Fortunately, IDE app stores like the Eclipse Marketplace [1] and Visual Studio Gallery [2] now allow users to seamlessly install third party extensions from their IDE. These IDE app stores have become popular, serving thousands of downloads per day. For researchers creating extensions as well as developers seeking to consume cutting edge ideas, the previously high cost of packaging, installation, and deployment has been dramatically reduced.

II. Choosing Your Battles: Which Research is Ripe for Transfer

Due to the reduced cost of deploying new research as extensions, more and more researchers are targeting practical, relevant problems. Thus, many new technologies may eventually be transferred into practice, but which ones are ready for transfer now? In our experience, there are two factors that lead to a technology being ready to transfer.

First, the technology must be an obvious improvement over the state-of-the-practice, as incremental improvements are not valuable enough to motivate developers. In our experience with Sando [1] the ability to search code using Google-style queries [2], formerly impossible, was a clear improvement. Similarly, developers utilizing NCrunch's continuous testing extension [3], [4] claim that NCrunch's ability to drastically reduce their feedback loop was a clear improvement.

Second, the technology's improvement must be proportional to its impact on the developers' workflow. Take for example, Code Recommenders, an extension that provides guidance on how to use APIs effectively based on how others have used them [5], [6]. Its success, as it has served hundreds of thousands of queries [7], is partially based on how easily it integrates into the user's existing workflow; instead of displaying all completions alphabetically (see Figure 1) it only shows the most relevant items (see Figure 2), optimizing the common case while requiring no changes in the workflow of the user.

III. Preparing for the Offensive: Hardening the Research Prototype

One of the most challenging issues surrounding transferring research is hardening the prototype enough to cross the chasm, or the theoretical gap between different types of user groups [8]. While asking researcher colleagues to try a new extension is possible–they are often eager to try new technology first–making the jump to developer colleagues and especially to unfamiliar developers requires significant implementation effort. Fortunately there are several successful strategies, as evidenced by existing extensions, for both initial implementation and subsequent hardening stages. One strategy, successfully used to develop Sando to target early adopters, is to form an open source project and recruit both academic and industrial partners. Another, used by Dig et al. to contribute refactorings to Netbeans [9], [10], is to

[1]http://marketplace.eclipse.org/
[2]https://visualstudiogallery.msdn.microsoft.com/

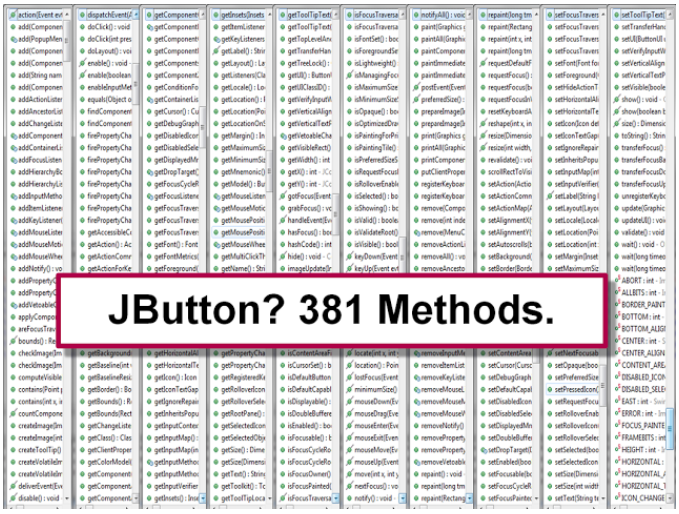

Fig. 1. The default Eclipse drop-down is a well-used feature, but certain objects can lead to overwhelmingly long lists.

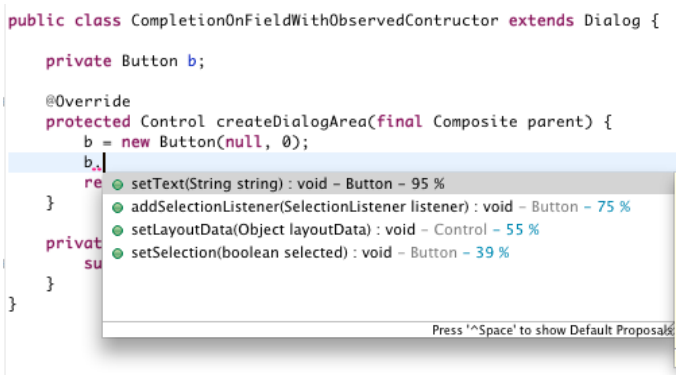

Fig. 2. Code Recommenders provides the same code completion experience (i.e., the drop-down), yet only the most relevant entries are shown.

piggyback onto an industrial group that is already creating related developer tooling. This strategy reduces the overall development effort for the researchers by leveraging the quality assurance and release engineering already in place. Finally another approach, as in the case of Mylyn [11], [12], Code Recommenders, and NCrunch, is to form a company that either fully or partially funds your development. We will discuss the advantages and disadvantages to each approach, as well as practical challenges that a technology transferring researcher will face, such as how to ensure the implementation continues even after the original grant funding is exhausted.

IV. Launching the Attach: Evangelizing and Popularizing Your Research

A common roadblock for researchers during tech transfer is their inexperience in evangelizing to developers. While proofs and in-lab experimental evaluations may convince academic colleagues software developers are both more skeptical and more pragmatic. To illustrate this point, a FindBugs [13] researcher once approached Eclipse committers, asking them "Did you know that there are X bugs in your component? I found them using FindBugs, a cutting-edge static analysis tool." Far from being impressed by an academic tool, the pragmatic committers replied, "Great! Did you file bugs so we can fix the issues?" Humbled, the researcher retreated. Yet he began to file bugs, and was able to convince developers that his technology worked on real code bases, eventually earning widespread adoption of his tool [14]. In this section of the talk we will discuss our own successes and failed efforts at evangelization, focusing on the differences between selling to researchers and evangelizing to developers. We will discuss, backed by download statistics, which strategies worked for Sando, such as technical blog posts. We will include key lessons learned including how developers react negatively to most marketing material yet are open to technical articles.

V. Speaker Biography: David Shepherd

David Shepherd, the primary speaker for this session, is an experienced practitioner who has been developing IDE extensions for over a decade. As employee #9 at Tasktop Technologies, Inc., a spin-off from University of British Columbia, he implemented and evangelized research-born technologies. As project lead on the open source Sando code search tool he transferred research conceived of in 2004, generating over 9,000 downloads and thousands of field usage reports. As team lead at ABB Corporate Research he oversees the design and implementation of extensions to improve the productivity and quality of ABB's codebase.

References

[1] D. Shepherd, K. Damevski, and B. Ropski. (2014) Sando's homepage. [Online]. Available: http://sando.codeplex.com
[2] A. Marcus, A. Sergeyev, V. Rajlich, and J. I. Maletic, "An information retrieval approach to concept location in source code," in *Working Conference on Reverse Engineering*. IEEE, 2004, pp. 214–223.
[3] R. Mulder. (2014) Ncrunch's homepage. [Online]. Available: http://www.ncrunch.net/
[4] D. Saff and M. D. Ernst, "An experimental evaluation of continuous testing during development," in *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4. ACM, 2004, pp. 76–85.
[5] M. Bruch, M. Monperrus, and M. Mezini, "Learning from examples to improve code completion systems," in *foundations of software engineering*. ACM, 2009, pp. 213–222.
[6] M. Bruch. (2014) Codetrail's homepage. [Online]. Available: http://www.codetrails.com/
[7] CodeTrails. (2014) Codetrail's usage homepage. [Online]. Available: http://download.codetrails.com/tracker/proposal-kind-duration/
[8] G. A. Moore, "Crossing the chasm," 2002.
[9] A. Gyori, L. Franklin, D. Dig, and J. Lahoda, "Crossing the gap from imperative to functional programming through refactoring," in *Foundations of Software Engineering*. ACM, 2013, pp. 543–553.
[10] Oracle. (2014) Netbeans's homepage. [Online]. Available: http://netbeans.org
[11] M. Kersten and G. C. Murphy, "Using task context to improve programmer productivity," in *Foundations of software engineering*. ACM, 2006, pp. 1–11.
[12] Tasktop. (2014) Tasktop's homepage. [Online]. Available: http://tasktop.com
[13] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler, "A few billion lines of code later: using static analysis to find bugs in the real world," *Communications of the ACM*, vol. 53, no. 2, pp. 66–75, 2010.
[14] B. Pugh and A. Loskutov. (2014) Findbug's homepage. [Online]. Available: http://findbugs.sourceforge.net/