A Case Study of Using HCI Methods to Improve Tools for Programmers

Andrew Faulring, Brad A. Myers
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{faulring, bam}@cs.cmu.edu

Yaad Oren, Keren Rotenberg
Technology & Innovation Platform BU
SAP Labs Israel Ltd.
Ra'anana, Israel
{yaad.oren, keren.rotenberg}@sap.com

***Abstract*—For more than five years, researchers at Carnegie Mellon University have been collaborating with several SAP teams to improve the usability of SAP's developer tools and programming APIs. Much research has shown that HCI techniques can improve the tools that developers use to write software. In a recent project, we applied HCI techniques to a SAP developer tool for the SAP NetWeaver Gateway product. The SAP team building this tool uses agile software development processes, which allowed them to quickly improve the tool's usability based upon the evaluations.**

***Keywords*-APIs; heuristic evaluation; cognitive walkthrough; agile software development; REST; OData; SAP NetWeaver Gateway; natural programming**

I. Introduction

The field of human-computer interaction (HCI) has developed a wide variety of techniques for the design and evaluation of user interfaces (UIs). The Natural Programming project [1] at Carnegie Mellon University (CMU) has repeatedly demonstrated that many of the these techniques can be adapted to improve the effectiveness of the tools and languages used by developers. For example, we have applied them to the user interfaces of integrated development environments (IDEs) [2, 3], debuggers [4], programming languages [5], and APIs (application programming interfaces) [6–10]. Many HCI techniques are applicable, and we have used contextual inquiry, heuristic evaluation, cognitive walkthrough, ethnography, and controlled lab studies. Improving the usability of developer tools can allow developers to work more efficiently and enable more people to become developers. This in turn should improve the quality of the resulting programs since more iterations can be made and more people can be involved in the development process.

For more than five years, our group at CMU has been collaborating with several teams at SAP to improve the usability of their developer tools. Members of our group have helped with the redesign of a business process rules engine called SAP BRFplus [11], and proposed improvements to SAP's eSOA (enterprise Service Oriented Architectures) web services [6, 10].

II. SAP NetWeaver Gateway

The SAP Business Suite provides customers with a set of business applications and tools for managing their supply chains, supplier relations, human resources, accounting, and so forth. SAP's software enables developers to create and customize new processes according to their company's business needs. SAP provides the core software, sometimes with industry-specific functionality, and then developers, who are often consultants, customize the software for each customer, often by writing code. SAP provides a range of programming and software management tools to support the customization process. Developers use SAP's ABAP programming language to write programs that interoperate with Business Suite systems.

Recognizing the growing market demand of exposing SAP services to non-SAP UI environments, such as mobile devices, the web, and social media applications, SAP created the NetWeaver Gateway [1] product, which provides a web services interface for applications to consume data stored inside a Business Suite system. SAP recognized that many mobile and web application developers were familiar with new web technologies such as REST [12], but did not have experience with SAP's ABAP language. So SAP developed the NetWeaver Gateway product to facilitate the consumption of SAP services by non-SAP developer communities. The NetWeaver Gateway provides an OData-based API for consuming services. OData [2] builds upon the HTTP REST architecture by adding a model for the data exchanged between client and server. Data is modeled as typed objects, called entities, such as a sales order or a customer. Entity sets are collections of entities of the same type, such as a set of sales orders. Links define relationships between entities, such as the customer of a sales order. OData uses the HTTP REST methods (GET, PUT, POST, DELETE) for exchanging the messages. There are OData libraries for common mobile programming languages (Objective-C, Java, .NET), and also for the Microsoft ASP.NET framework when an HTML-based user interface is desired. These programming languages enable applications to target the web and most mobile devices.

As part of the NetWeaver Gateway product, SAP developed the "SAP NetWeaver® Gateway developer tool for Visual Studio®" (hereafter "Gateway VS tool") that helps developers create "starter kit" applications for the ASP.NET framework. The Gateway VS tool provides a wizard-style user interface in which developers design a basic version of their application. The developer selects a Gateway service,

[1] http://www.sdn.sap.com/irj/sdn/gateway
[2] http://www.odata.org/

chooses the types of entities and the entity properties to be displayed, and specifies the navigation between entities. The Gateway VS tool then generates code, which implements the UI, and proxies, which connect to the SAP system. The developer can inspect and modify the generated code, and the code also serves as an example to help the developer learn how to consume Gateway services using the ASP.NET libraries. Similar tools and plug-ins for IDEs have also been developed for Eclipse and Adobe Flash Builder.

III. Usability Evaluations

The CMU team consulted for six months with the SAP NetWeaver Gateway Product Team. The CMU and SAP teams were not collocated, so we communicated electronically and held weekly teleconferences. As part of our consultation, SAP provided to us an early version of the Gateway VS tool, which had not yet been released to the market, and we performed two evaluations to find usability problems. In the first evaluation, we performed a heuristic evaluation [13] of the Gateway VS tool. In the second evaluation, we performed a cognitive walkthrough [14] of the Gateway VS tool by building an application for a common business use case that SAP had identified.

A. Heuristic Evaluation

The heuristic evaluation technique [13] helps to identify usability problems by having an expert examine the user interface with respect to heuristics that describe high-level behaviors that applications should have. Nielson developed an initial set of heuristics such as "speak the user's language: … the terminology in user interfaces should be based on the users' language and not on system-oriented terms" (p. 123 in [13]) and provide good error messages that use clear language and "constructively help the user solve the problem" (pp. 142–3 in [13]). We examined the tool's user interface to find where it might violate the heuristics. We identified usability gaps relating to the button labels, screen organization, error messages, inconsistent workflow, lack of warning messages, and so forth. We discussed these issues during our weekly teleconferences and later summarized all the issues in an internal report documenting the problems and suggesting ways to improve the user interface.

The SAP team developing this software uses the agile software development process[3]. The quick iteration cycle allowed them to improve the software based upon our feedback. We were often given new versions to evaluate that incorporated changes in response to issues we had recently reported. After about three months from when we delivered our report, SAP released an updated version of the Gateway VS tool to market, which incorporated several of our suggested changes.

The heuristic evaluation that was conducted by the CMU team provided valuable input to the SAP NetWeaver Gateway Product team from the point of view of a non-SAP developer who would like to interact with SAP services from their own native environment. The CMU team's internal report revealed how ASP.NET developers might approach the Gateway VS tool and what their expectations might be about how the user interface should work.

The SAP NetWeaver Gateway Product Team reviewed the internal report and prioritized the issues that it raised into different severity levels. As is accustomed in agile software development methodologies, there was a clear need to turn the evaluations into product specifications of tangible features for the next release. Hence, the CMU findings that were identified as "important" and "very important" were added to the next Gateway VS tool release development plan.

B. Cognitive Walkthrough

In contrast to heuristic evaluation, which takes a more holistic view, the cognitive walkthrough technique evaluates how well a user interface supports one or more specific tasks [14, 15]. The SAP team developed a common business use case for a typical application that would consume SAP NetWeaver Gateway services. The CMU team then attempted to build such an application using the Gateway VS tool and the documentation provided by SAP. As we worked to build the application we carefully documented the issues that we encountered. For this evaluation, we used the updated Gateway VS tool, which had already fixed several of the important problems that we had reported based on our heuristic analysis.

This evaluation focused on modifying the "starter kit" application code that the Gateway VS tool generates. We discovered several problems with the generated code. Since the NetWeaver Gateway was still under development, this is not unexpected. While we were able to work around the bugs in the generated code, we encountered problems with the server that prevented us from completing the common business use case. We also received error messages that we did not fully understand and found a few additional issues with the user interface of the tool itself. We discussed these problems during the weekly teleconferences and summarized them in a final report, which carefully documented the usability problems and functional errors with the Gateway VS tool, the generated code, and the server.

Just as in the previous phase, SAP's use of the agile software development method enabled them to fix the bugs and provide us with updated versions of both the Gateway VS tool and the Gateway server. We were then able to continue to work on the use case from the point where we had previously become stuck. With the bugs fixed, we were able to complete the common business use case and write a summary report. In this final report, we noted some of the difficulties we had modifying the code due to our lack of familiarity with the ASP.NET libraries. Since SAP plans to target this tool to developers who do have experience using that library, our findings highlight the need for the tool's documentation to clearly state what pre-requisite knowledge a developer should have and to provide suggestions for obtaining information about third-party tools when needed.

IV. Agile Development and HCI Techniques

As have others [16], we found the agile development and HCI techniques to work well together. The agile process splits the development effort into a series of *sprints*, each

[3] http://www.agilealliance.org/

lasting a pre-defined number of weeks. Each sprint begins with a planning meeting in which the product owner explains the product requirements to the team. The team then drills into each requirement to create tasks and provide effort estimates, and reports back to the product owner to commit to the requirements that can fit into the sprint. The HCI evaluation techniques can be used to quickly evaluate the latest version of the software, generating requirements to be addressed in upcoming sprints. The rapid, iterative nature of both these processes work well together.

The sprint ends in a review meeting where the team shows the product owner the improvements made during the sprint. In addition, the product owner can decide after the review meeting to release the product to the market or only to small a group of people, such as to the CMU evaluators, according to the product's stability, maturity, and so forth.

V. Conclusions

Our collaboration has demonstrated that HCI techniques can be applied to improve the usability of software development tools. When combined with agile software development methodology, HCI techniques can yield quick improvements to such tools.

SAP NetWeaver Gateway Product team has been constantly engaging with different non-SAP development communities in order to create a product that makes it easier to consume the SAP services from any UI environment. The collaboration with CMU had corroborated the importance of evaluating how non-SAP developers would interact with SAP services and of identifying the gaps in expectations, especially when compared to a more SAP-savvy developer. The applicability of CMU findings extends beyond the Gateway VS tool. Based on the reports, several product specifications were also incorporated for the next release of SAP NetWeaver Gateway plug-in for Eclipse. In October 2011 SAP NetWeaver Gateway, including the developer tools, was released for general availability in the market.

Looking into the future, the SAP NetWeaver Gateway Product team has been inspired to continue to utilize human-centered approaches when creating tools for software developers, through close engagement with the different developer communities. This will enable those developers to consume services from their SAP systems in an easy-to-use, controlled, and effective way. Meanwhile, the CMU team is now collaborating with a different group at SAP, to further evaluate and adapt HCI methods to improve many different forms of APIs and programmer tools.

Acknowledgment

We thank SAP Labs for funding this research, in particular Ralf Ehret, Paul Hofmann and Ike Nassi. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of SAP.

References

[1] B. A. Myers, A. J. Ko, and M. M. Burnett, "Invited research overview: end-user programming," in *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI'06)*, Montreal, Canada, 2006, pp. 75–80.

[2] M. Mooty, A. Faulring, J. Stylos, and B. A. Myers, "Calcite: completing code completion for constructors using crowds," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'10)*, Leganés-Madrid, Spain, 2010, pp. 15–22.

[3] T. D. LaToza and B. A. Myers, "Visualizing call graphs," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'11)*, Pittsburgh, PA, 2011, pp. 117–124.

[4] A. J. Ko and B. A. Myers, "Debugging reinvented: asking and answering why and why not questions about program behavior," in *Proc. ACM/IEEE Int. Conference on Software Engineering (ICSE'08)*, Leipzig Germany, 2008, pp. 301–310.

[5] J. F. Pane, B. A. Myers, and L. B. Miller, "Using HCI techniques to design a more usable programming system," in *Proc. IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02)*, Arlington, VA, 2002, pp. 198–206.

[6] J. Beaton, S. Y. Jeong, Y. Xie, J. Stylos, and B. A. Myers, "Usability challenges for enterprise Service-Oriented Architecture APIs," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'08*), Herrsching am Ammersee, Germany, 2008, pp. 193–196.

[7] D. S. Eisenberg, J. Stylos, A. Faulring, and B. A. Myers, "Using association metrics to help users navigate API documentation," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'10)*, Leganés-Madrid, Spain, 2010, pp. 23–30.

[8] J. Stylos, A. Faulring, Z. Yang, and B. A. Myers, "Improving API documentation using API usage information," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'09)*, Corvallis, OR, 2009, pp. 119–126.

[9] J. Stylos, B. A. Myers, and Z. Yang, "Jadeite: improving API documentation using usage information," in *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI'09)*, Boston, MA, 2009, pp. 4429–4434.

[10] B. A. Myers, S. Y. Jeong, Y. Xie, J. Beaton, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K. Busse, "Studying the documentation of an API for enterprise Service-Oriented Architecture," *JOEUC: The Journal of Organizational and End User Computing*, vol. 22, pp. 23–51, Jan–Mar 2010.

[11] J. Stylos, D. K. Busse, B. Graf, C. Ziegler, R. Ehret, and J. Karstens, "A case study of API design for improved usability," in *Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC'08)*, Herrsching am Ammersee, Germany, 2008, pp. 189–192.

[12] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Information and Computer Science, Univ. of California, Irvine. 2000.

[13] J. Nielsen, *Usability Engineering*. Boston, MA: Academic Press, 1993.

[14] M. H. Blackmon, P. G. Polson, M. Kitajima, and C. Lewis, "Cognitive walkthrough for the web," in *Proc. ACM Conference on Human Factors in Computing Systems (CHI'02)*, Minneapolis, MN, 2002, pp. 463–470.

[15] C. Lewis, P. G. Polson, C. Wharton, and J. Rieman, "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces," in *Proc. ACM Conference on Human Factors in Computing Systems (CHI'90)*, Seattle, WA, 1990, pp. 235–242.

[16] M. Budwig, S. Jeong, and K. Kelkar, "When user experience met agile: a case study," in *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI'09)*, Boston, MA, 2009, pp. 3075–3084.