

Cognitive Walkthrough + Personae = An Empirical Infrastructure for Modeling Software Developers

R. Naghshin
HCSE Group
Concordia University,
r_naghsh@cs.concordia.ca

A. Seffah
HCSE Group
Concordia University,
seffah@cs.concordia.ca

R. Kline
HCSE Group and Psychology
Concordia University,
rbkline@vax2.concordia.ca

Abstract

This is the third in a series of reports about usability and learnability problems of integrated development environments (IDE). The first report used the method of ethnographic interviews, and the second report the methods of heuristic and psychometric evaluation to study problems that developers face in using IDEs. The present study extends previous work by applying the method of direct behavioral observation of more versus less experienced users of the same IDE for C++. We demonstrate in this paper how one can effectively use cognitive walkthroughs and code walkthroughs for empirically modeling developers' characteristics and personae.

1. Introduction

Participatory design and user-centered approaches to software development generally aim to (1) involve users earlier in the design process, and (2) help programmers better understand the needs of end-users. Examples of such techniques include scenarios [1], cognitive walkthroughs (CogW) that asks users to perform certain tasks with an application, [2] and the construction of a persona that describes end-users for the benefit of application developers [3]. User-centered methods like those just described have not typically been applied in the development of IDEs where the end-users are software developers. This is unfortunate because there is simple evidence of usability and learnability problems with IDEs [e.g., 4-6].

Part of the problem is that the user interfaces (UI) of IDEs are often quite complex, and functionalities in these UIs may be presented in non-intuitive ways. This makes many IDEs unnecessarily difficult to learn. Another problem is that the same IDE may be used by developers with different backgrounds and levels of experience. This makes it more difficult to

design an IDE that will suit the needs of heterogeneous groups of developers.

This present work is the third in series of studies that have used increasingly specific method to study usability and learnability shortcomings of IDEs. The first study [5] used ethnographic interviews of programmers to identify general problems in their use of IDEs for Java, and the second [6] used heuristic evaluation and psychometric assessment with users of IDEs for C++. These studies found quite similar usability problems across these different IDEs.

This study uses direct observational methods with aspects of CogW and code walkthrough (CodW) as users of an IDE for C++ attempt to solve different kinds of problems with it. The observational are then synthesizes to find potential usability, learnability and comprehension programs. User behaviors while interacting with the IDE were also monitored screened for patterns that may contribute to this understanding. At the end of the paper we discuss how all of this information may be combined to construct personae of developers. These representations may help software engineers who develop IDEs better understand the needs of software engineers who use them.

2. Method

2.1 Participants

A total of 6 developers participated in this study. They were all advanced computer science graduate students with similar academic in C++ programming. A total of three participants were very experienced with C++ IDE used in this project, Microsoft Visual C++ (Version 6). The other three were less experienced with this particular IDE.

2.2 Procedure

Each participant was observed in a laboratory setting where their interactions with the IDE took place on a workstation in one room while the experimenter (R.N.) monitored the session on a separate workstation in a different room. A video camera and the Adobe Premier application was used to capture the participant's use the keyboard and mouse so that it could be observed by the experimenter. The Camtasia application and the Timbuktu Remote Control were used to record all participants' interactions with the IDE, including keyboard and mouse use for menu selection and the like. The total length of each observation session was about 30 minutes.

The participants were presented with a C++ program about an elevator simulation. This program consists of about 1,000 lines of code stored across 11 different files. They were asked to carry out five basic tasks common to basically any IDE for program development, including the creation of a new project, editing and comprehension of source code, compilation of a whole or partial application, debugging, and archiving the generated application and source code. For example, the source code contained a bug that prevented the elevator from traveling to certain floors. The participants were asked to find and correct this bug and recompile the modified source code. This part of the study is similar a CodW in which potential defects in source must be found and corrected.

3. Results

3.1 Observational Results

The following usability, learnability, and comprehensions problems were apparent based on review of the behavioral data across all 6 participants:

1. The occasional users rarely used shortcuts like displaying properties of program artifacts by clicking the right mouse button. They seemed to be unaware of such shortcuts, and the IDE did nothing to make them more obvious.
2. All occasional users had difficulties creating a new project. They often started by simply opening the main file without specifying a new project. Their attempts to subsequently compile modified source code caused linking errors. Although this is a common

kind of mistake, the IDE did nothing to prevent it.

3. Occasional users did not always understand that the same function was represented in different ways in the text-based menu bar and the icon-based toolbar until they tried it both ways.
4. Both experienced and occasional users had trouble understanding error messages during compilation of the source code. Specifically, they found the language in these messages to be overly technical and sparse.
5. All users also had difficulty with attempting to archive the modified application by saving files under different names or directories. The occasional users in particular were often uncertain whether these functionalities are part of the IDE or required looking outside it to operating system tools, such as a file/directory explorer.

3.2 From Observations to Personae

The idea of constructing personae to describe a target clientele has been part of marketing studies for some time [7]. The goal is to create a set of personae that represents best the target audience, in this case more experienced versus more occasional users of an IDE for C++. Program managers may be included in the latter category.

In our observational framework, the methods of CogW and CodW are applied in an iterative process to construct personae of software developers, which is illustrated in Figure 1.

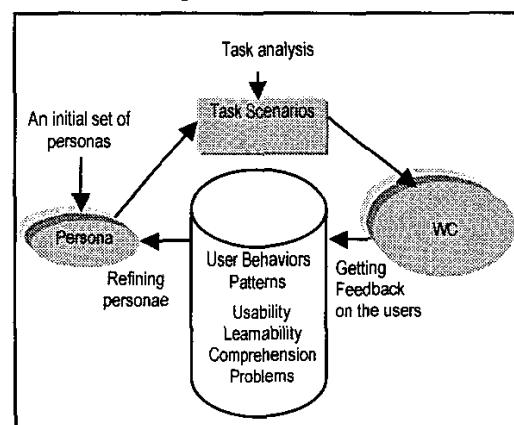


Figure 1: Proposed iterative framework for eliciting and validating Personae via cognitive and code walkthrough studies.

We started the process by creating a pair of personae for Microsoft Visual C++ that represents two different types of users, occasional versus regular. The persona for the regular user is that of "Jim Smith," a 24 year-old masters student who is also employed as a programmer. The persona for the occasional user is that of "Charles Butler," a 46 year-old project manager who uses Visual C++ mainly to check the work of programmers who work under him.

We tried to refine the pair of personae described above by examining the observational data for the regular versus occasional users of Visual C++ who participated in this study. The personae descriptions after incorporating the observational results are presented in Table 2. Briefly, these personae include some biographical information to make them more realistic, but most of the descriptions concern preferred means of program interaction.



 <p>Jim Smith</p> <ul style="list-style-type: none"> • 24-year-old Masters student in computer engineering, Internet and gaming addict. • Works in Microsoft as a C++ programmer. He loves his job and he loves playing with the software and discovering features that not everyone knows. • Jim knows Visual C++ has a lot of functions, and he checks computer magazines to see if there is any new critical function in the new version. • He is very impatient. When compiling takes long he loses his patience and may restart the computer. 	 <p>Charles Butler</p> <ul style="list-style-type: none"> • 46-year-old project manager of a small-sized software • He knows programming in different languages, and the worst thing anyone can tell him is he is not fast enough. This is because he believes a project manager should know these kinds of things. • Thomas wants to use only the basic functions. He believes that about 60% of the functionalities in Visual C++ are there for the sake of marketing. • He hates long error messages. • He wants a hierarchical view of the program in order to understand the project without having to ask people who work under him.
--	---

Table 2. Revised Set of Personas for Microsoft Visual C++

We plan to pursue our investigations by considering a larger and different class of personae. We are implementing a Web-based system for collecting

information on software developers' with different background and skills and working in different environments. Such information will be used to create "provisional personae," which are a sketchy best guess at user needs and characteristics. They typically consist of a few goals and one or two other characteristics, but no detail or narrative. It is important to notice that provisional personae are useful thought experiments, but are not real personae because they are not based on empirical data. Further research has to be done on using persona descriptions in the user-design process of human-centric CASE tools.

References

- [1] Bodker, S. (2000). Scenarios in user-centered design: Setting the stage for reflection and action. *Interacting with Computers*, 13(1), 61-75, 2000.
- [2] Wharton, C., Rieman, J., Lewis, C., and Polson, P. (1994). The cognitive walkthrough method: A practitioner's guide. In J. Nielsen and R.L. Mack (Eds.), *Usability inspection methods* (pp.105-141). New York: John Wiley.
- [3] Cooper, A. (1999). *The inmates are running the asylum: Why high tech products drive us crazy and how to restore sanity*. New York: Macmillan.
- [4] Ivori, J. (1996). Why are CASE tools not used? *Communication of the ACM*, 39(10), 94-103.
- [5] Kline, R., Seffah, A., Javahery, H., Donayee, M., & Rilling, J. (2002, September). Quantifying developer experiences via heuristic and psychometric evaluation. *Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments*, Arlington, VA, pp. 34-36.
- [6] Seffah A., & Rilling, J. (2001, September). Investigating the relationship between usability and conceptual gaps for human-centric CASE tools. *IEEE Symposium on Human-Centric Computing Languages and Environments*, Stresa, Italy.
- [7] Lewis, C., & Rieman, J. (1993). Task-centered user interface design: A practical introduction. Distributed via anonymous ftp (Internet address: ftp.cs.colorado.edu), *International Journal of Man-Machine Studies*, 36, 741-773.