Chapter 4.16
Patchwork Prototyping with Open Source Software

M. Cameron Jones
University of Illinois at Urbana-Champaign, USA

Ingbert R. Floyd
University of Illinois at Urbana-Champaign, USA

Michael B. Twidale
University of Illinois at Urbana-Champaign, USA

ABSTRACT

This chapter explores the concept of patchwork prototyping: the combining of open source software applications to rapidly create a rudimentary but fully functional prototype that can be used and hence evaluated in real-life situations. The use of a working prototype enables the capture of more realistic and informed requirements than traditional methods that rely on users trying to imagine how they might use the envisaged system in their work, and even more problematic, how that system in use may change how they work. Experiences with the use of the method in the development of two different collaborative applications are described. Patchwork prototyping is compared and contrasted with other prototyping methods including paper prototyping and the use of commercial off-the-shelf software.

INTRODUCTION

The potential for innovation with open source software (OSS) is unlimited. Like any entity in the world, OSS will inevitably be affected by its context in the world. As it migrates from one context to another, it will be appropriated by different users in different ways, possibly in ways in which the original stakeholders never expected. Thus, innovation is not only present during design and development, but also during use (Thomke & von Hippel, 2002). In this chapter, we explore an emerging innovation through use:

a rapid prototyping-based approach to requirements gathering using OSS. We call this approach *patchwork prototyping* because it involves patching together open source applications as a means of creating high-fidelity prototypes. Patchwork prototyping combines the speed and low cost of paper prototypes, the breadth of horizontal prototypes, and the depth and high functionality of vertical, high-fidelity prototypes. Such a prototype is necessarily crude as it is composed of stand-alone applications stitched together with visible seams. However, it is still extremely useful in eliciting requirements in ill-defined design contexts because of the robust and feature-rich nature of the component OSS applications.

One such design context is the development of systems for collaborative interaction, like "cybercollaboratories." The authors have been involved in several such research projects, developing cyberinfrastructure to support various communities, including communities of learners, educators, humanists, scientists, and engineers. Designing and developing such systems, however, is a significant challenge; as Finholt (2002) noted, collaboratory development must overcome the "enormous difficulties of supporting complex group work in virtual settings" (p. 93). Despite many past attempts to build collaborative environments for scientists (see Finholt for a list of collaboratory projects), little seems to have been learned about their effective design, and such environments are notorious for their failure (Grudin, 1988; Star & Ruhleder, 1996). Thus, the focus of this chapter is on a method of effective design through a form of rapid, iterative prototyping and evaluation.

Patchwork prototyping was developed from our experiences working on cybercollaboratory projects. It is an emergent practice we found being independently redeveloped in several projects; thus, we see it as an effective ad hoc behavior worthy of study, documentation, and formalization. Patchwork prototyping is fundamentally a user-driven process. In all of the cases where we saw it emerge, the projects were driven by user groups and communities eager to harness computational power to enhance their current activities or enable future activities. Additionally, the developers of the prototypes had no pretence of knowing what the users might need a priori. As a result, patchwork prototyping's success hinges on three critical components:

1. Rapid iteration of high-fidelity prototypes
2. Incorporation of the prototypes by the end users into their daily work activities
3. Extensive collection of feedback facilitated by an insider to the user community

In this chapter, we focus on how the method worked from the developers' point of view. It is from this perspective that the advantages of using OSS are most striking. However, one should bear in mind that the method is not just a software development method, but also a sociotechnical systems (Trist, 1981) development method: The social structures, workflows, and culture of the groups will be coevolving in concert with the software prototype.

REQUIREMENTS GATHERING IN COLLABORATIVE SOFTWARE DESIGN

Software engineering methods attempt to make software development resemble other engineering and manufacturing processes by making the process more predictable and consistent. However, software cannot always be engineered, especially Web-based applications (Pressman et al., 1998). Even when application development follows the practices of software engineering, it is possible to produce applications that fail to be used or adopted (Grudin, 1988; Star & Ruhleder, 1996). A major source of these problems is undetected failure in the initial step in building the system: the requirements-gathering phase. This is the