

Comparing Procedural and Object-oriented Design

Adrienne Lee and Nancy Pennington

University of Colorado

Psychology Department

Boulder, CO 80309-0345

(303)492-5394, lee_a@clipr.colorado.edu

(303)492-6655, npennington@clipr.colorado.edu

Scott Wolff

U S WEST Advanced Technologies

4001 Discovery Drive

Boulder, Colorado 80303

(303)541-6302, scott@uswest.com

INTRODUCTION

Many companies employ programmers who use traditional procedural methods for software design. A new approach, object-oriented design, which allows for easy extensibility and reuse of previous designs, has recently been developed (Wirfs-Brock, Wilkerson, & Wiener, 1990). Major claims are made about the benefits of object-oriented design (Gibson, 1990). However, anecdotal evidence indicates that people who know procedural design have difficulty learning object-oriented design. Unfortunately, the benefits of object-oriented design will be lost if programmers have problems switching paradigms.

The general features and characteristics of procedural design are well documented (Adelson & Soloway, 1988; Atwood & Jeffries, 1980; Guindon & Curtis, 1988). In contrast, very few studies have examined object-oriented designers. Most comparisons between the two design paradigms (except Rosson & Gold 1989) have been informal (Korson & McGregor, 1990; Rosson & Alpert, 1990).

METHOD

The purpose of this study was: 1) to compare expert object-oriented and procedural designers executing a design for the same problem, and 2) to identify problems in transfer for object-oriented design novices who have extensive procedural experience.

Subjects. Design protocols were collected from 5 expert object-oriented designers, 5 expert procedural designers, and 5 novice object-oriented designers with procedural backgrounds.

Materials. Subjects designed a scoring system for swim meets (problem adapted from Rumbaugh, Blaha, Premerlani, Eddy & Lorensen, 1991).

Procedure. Talk-aloud protocols were collected from designers as they worked on their designs. Experts were asked to specify a target language for which they felt most comfortable; however, novice object programmers were asked to design using an object-oriented language. The sessions were videotaped.

Analysis. Videotapes were transcribed and annotated to include diagrams and notes made during the design process. Designs were then characterized in terms of decompositions, relations specified among design subparts, design decisions explicitly made, specific use of existing code or other resources, and the manner in which design evaluations were conducted.

RESULTS/DISCUSSION

Research has shown that procedural designers reuse parts of their designs when creating a new design (Guindon, 1990). In our study, object-oriented designers referred to libraries of functions that they could employ, while procedural designers referred to classes of designs (e.g. "this is a database design"). So, although reuse may be important for both types of designers, for object-oriented designers, reuse appeared to extend beyond familiar previous designs because they had easy memory access to the code libraries. Both object-oriented and procedural designers mentioned ways in which they designed their systems to be extensible. However, object-oriented designers produced more closely matching designs (similar objects and methods) which would allow other object-oriented designers to understand the design and extended it more easily. There was no evidence to support the idea that object-oriented design is easier than procedural design, at least for the problem we studied. In general, object-oriented designers took longer and needed more drawing space to work out ideas.

Many characteristics of design were general to both paradigms, such as creating drawings, considering different design possibilities, and checking the design periodically. Several differences between object-oriented and procedural design were noted. First, procedural designers tended to focus on the interface, input and output, data and control flow. However, object-oriented designers focused on how each object maintained its own data and calculations. Second, procedural designers focused on single rather than distributed action. Third, object-oriented designers used anthropomorphic phrases and viewed the domain in terms of active agents.

Novice object-oriented designers were able to produce designs similar to the experts. One problem novices had was avoiding focus on I/O, data and control flow, especially in the initial stages of problem solving. A second problem was that novices initially chose too many objects and worked harder in the elimination process. In addition, they had more difficulty than the experts with how objects were connected in a class hierarchy.

This research compared procedural and object-oriented designers, described some differences between the paradigms and some difficulties novices with a procedural background encounter. Differences do not indicate which paradigm is better but they provide insight into instructional concerns in retraining for object-oriented design.