Reusability is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required. Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers. Proponents claim that objects and software components offer a more advanced form of reusability, although it has been tough to objectively measure and define levels or scores of reusability. Reusability implies some explicit management of build, packaging, distribution, installation, configuration, deployment, maintenance and upgrade issues. If these issues are not considered, software may appear to be reusable from design point of view, but will not be reused in practice. This paper presents an empirical study of the
software reuse activity by expert designers in the context of object-oriented design. Our study focuses on the three following aspects of reuse: (1) the interaction between some design processes, e.g. constructing a problem representation, searching for and evaluating solutions, and reuse processes, i.e. retrieving and using previous solutions, (2) the mental processes involved in reuse, e.g. example-based retrieval or bottom-up versus top-down expanding of the solution, and (3) the mental representations constructed throughout the reuse activity, e.g. dynamic versus static representations.

Reference


Index Terms

Computer Science
Software Engineering

Key words

Reusability
object-oriented design