CONTRIBUTION TO: PANEL ON OBJECT-ORIENTED KNOWLEDGE-BASED SOFTWARE MAINTENANCE PLATFORMS, CHAIRPERSON: R. MIKKILINENI, OOPSLA'88

Software Maintenance Environments: a New Perspective

Gerhard Fischer, Roger King, Gary Nutt, Leon Osterweil and Christian Rathke
Department of Computer Science, University of Colorado, Boulder

Software environments of the future have to support design methodologies whose main activity is not only the generation of new independent programs but also the maintenance, integration, modification, and explanation of existing ones. Our joint research tries to find answers to this challenge in an interdisciplinary research project bringing together researchers from software engineering, databases, human-computer communication, knowledge representation and system modeling. Some of the major research themes pursued are: large-grain object-oriented environments for software engineering, process programming, visualization of complex processes, human problem-domain communication and reuse and redesign.

Reuse and redesign can be efficiently supported by object-oriented construction kits. Construction kits are based on a number of abstractions that characterize certain domains (e.g., user interface design, network design, task modeling) -- and are able to support "human problem-domain communication". The abstractions for a domain constitute a partial "theory" of a class of software systems. The evolutionary development of such a framework, driven by testing the validity of abstractions in a variety of different applications, is a prerequisite for a system to support reuse and redesign.

The limited success of reuse and redesign as a major programming methodology is in our opinion directly related to the lack of adequate support tools. Having a large set of building blocks available without good retrieval tools is a mixed blessing. The advantage is that, in all probability, an existing building block or set of building blocks -- which have been used and tested before -- either fit the users' needs directly or come close to doing so. The disadvantage is that it may take a long time to discover a suitable building block or to find out whether it exists at all. Using a high functionality computer system in software design tasks reduces the size of the application system substantially because the system designer can take advantage of the abstractions available in the basic system. The major costs incurred by the system designer in using high functionality systems is in learning and understanding the abstraction space offered -- but the designer incurs these costs only once.

In our joint research project, we are integrating the following prototypical systems components (constructed independently in other research efforts) into c<Floss>, a large-grained, object-oriented architecture for software environments: OBJTALK -- an object-oriented knowledge representation language; CACTIS -- an object-oriented, distributed database; OLYMPUS -- a modeling system to animate and simulate graphs; NEWTON -- a debugging system with a user interface developed in WLISP (a user interface toolkit) and NODE -- a graphical editor design environment incorporating knowledge about application domains.

Acknowledgments. Our research is supported by a grant No. 0487.12.0389B from UsWest Advanced Technologies.