Composing Systems Using Tool Abstraction

David Garlan *
Tektronix Laboratories
P.O. Box 500, MS 50-662
Beaverton, OR 97077

Gail E. Kaiser †
Columbia University
Department of Computer Science
New York, NY 10027

David Notkin ‡
University of Washington
Department of Computer Science & Engineering, FR-35
Seattle, WA 98195

November 10, 1990

Abstract

The well-known notion of data abstraction, which separates the interface of a module from its implementation, permits the representation and implementation of abstract data types to evolve without affecting the module's clients. However, many enhancements change the function of a system, not just its implementation. Data abstraction does not directly ease the difficulties and costs of such enhancements. *Tool abstraction* complements data abstraction by supporting evolutionary changes that are made incrementally and independently. This approach structures a system as a pool of shared data structures, viewed by a collection of cooperating tools, each of which provides a piece of the overall function. This article identifies tool abstraction as the common foundation of a variety of existing systems — including spreadsheets, structure-oriented environments, production systems, and object-oriented systems.

©1990 D. Garlan, G. E. Kaiser and D. Notkin

Keywords: Tool abstraction, data abstraction, evolution, modularization, tool integration.

*Garlan's address is now Carnegie Mellon University, School of Computer Science, Schenley Park, Pittsburgh, PA 15213.

†Kaiser is supported by National Science Foundation grants CCR-9000930, CDA-8920080 and CCR-8858029, by grants from AT&T, BNR, Citicorp, DEC, IBM, SRA, Sun and Xerox, by the Center for Advanced Technology and by the Center for Telecommunications Research.

‡Notkin is supported in part by National Science Foundation grant CCR-8858804, Air Force Office of Scientific Research contract AFOSR-88-0023, and grants from DEC and Xerox.