UNIVERSITY OF ADELAIDE LIBRARY
22-9-95

Program Construction and Evolution in a Persistent Integrated Programming Environment

Alex Miklós Farkas

Thesis submitted for the degree of
Doctor of Philosophy
in
The University of Adelaide
(Faculty of Engineering)

September 1995

Abstract

In orthogonally persistent systems, data is treated independently of its persistence allowing storage management details to be ignored by programmers. The coexistence of all data within a persistent repository provides new opportunities for the persistent environment to participate in a number of software engineering processes, namely, software:

- construction,
- debugging,
- maintenance,
- evolution, and
- distribution.

A novel programming methodology supporting various styles of binding allows the persistent environment to participate in the software construction process. Existing programs and data may be bound to an application at different times during the application's lifetime, namely, at run-time, compile-time and program construction-time. This wide range of binding styles enables the programmer to trade off binding safety and binding flexibility.

A mechanism called Octopus enables the programmer to access a rich source of information about the bindings within arbitrary values. Using this information, existing applications may be interrogated and manipulated in ways in which the programming language alone would not normally permit. However, Octopus ensures that the type system of the programming language remains inviolate, thus providing a type safe mechanism for constructing debugging, maintenance and evolutionary tools. Another feature of Octopus is that it permits portions of an application's object graph to be isolated and transmitted.

Nodules permit a style of template programming that enables data values and executable code to be shared between different Nodule instances. The essence of this mechanism is to allow programs to be constructed and compiled without the requirement that the values used by the program be present. In this manner, individual components may be constructed independently and later assembled to form a complete application. Nodules also provide a

rich source of information about the structure of an application and may be used in conjunction with Octopus to develop and maintain software components.

This thesis describes in detail a persistent programming environment, the Octopus and Nodule mechanisms, and illustrates how each of these mechanisms, when used in conjunction with each other, provide a persistent integrated programming environment that supports a wide variety of software engineering activities otherwise lacking in conventional, or non-persistent systems.

Contents