Creating an Open Environment Software Infrastructure

by

N93-16774

Michael J. Jipping
Assistant Professor
Hope College
Department of Computer Science
Holland, MI 49423

Abstract

As the development of complex computer hardware accelerates at increasing rates, the ability of software to keep pace is essential. The development of software design tools, however, is falling behind the development of hardware for several reasons, the most prominent of which is the lack of a software infrastructure to provide an integrated environment for all parts of a software system. The research undertaken by the author at NASA LaRC in the summer of 1992 has undertaken to provide a basis for answering this problem by investigating the requirements of open environments.

1 Introduction

With the rapid development of digital processing technology, NASA programs have become increasingly dependent on the capabilities of complex computer systems. Current flight control research, for example, advocating active controls and fully integrated guidance and control systems , relies heavily on digital processing technology. These advanced guidance and control systems, designed to optimize aircraft performance, will demand high-throughput, fault-tolerant computing systems. The functional performance, reliability, and safety of these systems are of great importance to NASA and thus research within NASA's Aeronautics Controls and Guidance Program is directed toward the development of design, assessment, and validation methodologies for flight crucial systems.

The state-of-the-art of this technology, however, is reflected in a primary issue resulting from a NASA-LaRC workshop on digital systems technology i.e. " lack of effective design and validation methods with support tools to enable engineering of highly-integrated, flight-critical digital systems". Design methods are generally fragmented and do not support integrated performance, reliability, and safety analysis and there is a growing recognition that such integrated studies will require an integrated design and evaluation environment .

The research focus of the Systems Architecture Branch of NASA-LaRC is the Automated Design Technology (ADT) for engineering safety-critical software and architecture systems for advanced aircraft avionics. This work is motivated by the belief that focused research on application-specific domains will result in significant gains in productivity and quality. The need for such research was also recommended in a 1989 National Research Council Computer Science and Technology Board workshop – "it is critical to recognize the legitimacy of specialization to the domain at the expense of expressive generality".

2 The Project

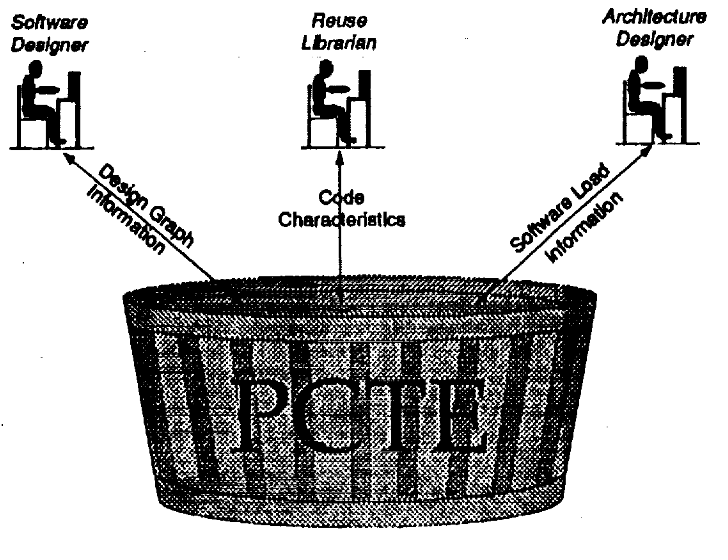
In order to support ADT, a project was initiated to construct and evaluate an open environment software infrastructure as the framework for this design technology. The environment was based on the Integrated Project Support Environment model of software integration, where a complete infrastructure is build in which software tools are embedded. Further, the environment was built in accordance with the Portable Common Tool Environment standard of IPSE systems. The main advantages of using PCTE are many; the biggest advantages are (1) a common data infrastructure were all data is viewed as objects and resources are provided to all program in the environment to manipulate those objects, and (2) all objects are highly organized by relationships to each other, thus enabling easy reference and access to any object in the system.

The project, shown in Figure 1, was undertaken to show the advantages of PCTE in practical terms. In this scenario, we have a three engineers working in a PCTE environment. The software engineer is designing software, which is being cataloged by the reuse librarian and being executed on a system designed by the architecture engineer. The software engineer sees her software in terms of control and data flow graph information; the reuse librarian see coding characteristics; and the architecture designer sees the load information the software will put on the hardware he is designing. Each engineer has a perspective of the software being design; it is the software infrastructure that allows these perspectives to be integrated into one software model. Our scenario combined three "real" programs jointly developed with NASA LaRC: CASE for software design, InQuisiX for the reuse librarian, and ADAS for the hardware design tool.

PCTE allows objects and their manipulation to be hidden from the user and her software. Each user's software accesses what it should in the correct format and translations or information derivation is under the surface of the IPSE.

It is hoped that the construction and demonstration of this software infrastructure will inspire others to use an IPSE to design and implement their software systems. Examples are numerous: different, off-the-shelf geographic modelling programs implementing their own GIS models as *perspectives* of some common geographic object; spreadsheets from different vendors on different computers sharing complex information on mission launches as if they were in the same format; database systems accessing each others information.

Figure 1: The PCTE-based Demonstration Environment


Software Designer
Reuse Librarian
Architecture Designer
Design Graph Information
Code Characteristics
Software Load Information
PCTE