Potential use of the Object Paradigm for Software Engineering Environments in the 1990s

Rao V. Mikkilineni
Director, Advanced Software Technologies
U S WEST Advanced Technologies
6200 South Quebec
Englewood, Colorado 80111

Abstract

Recent advances in programming languages, database technology, expert systems, and distributed workstations hold promise for improving software productivity and quality in the future. Object-oriented programming allows building software that is easy to change and reuse. Object databases are attempting to provide management of data involved in the software life-cycle (e.g., project details, requirements, design documents, CASE tools, software documentation, and so forth). Expert systems are attempting to automate some aspects of the software life-cycle (e.g., project management, source code control, automated make, so forth). Distributed workstations, with CASE tools and local processing power, are increasing programmer productivity.

Based on these advances, researchers are working on a new data model that will combine the object-oriented language features, database features, inferencing, and distribution. This will allow new software environments that are even more efficient because of the seamless integration of these technologies. This will expedite future software prototyping, production, and maintenance.

Introduction

The software bottleneck problem is well documented in literature [1, 2]. While there has been progress in developing mass-produced software, we are still not able to meet the rising demand for custom software. Recent advances in several emerging technologies are promising to provide some relief. These include object-oriented languages, databases, expert systems, and distributed workstation environments.

Although there is no single, silver bullet [3], a combination of these remedies will provide powerful environments that were not possible before. In addition, the confluence of these advances, as they evolve, will create a new data model that will merge the essence of these technologies. The synergy obtained by such a seamless integration will produce a next generation software engineering environment that is more efficient than today's environment. Figure 1 depicts these emerging technologies.

Object programming bases the modular decomposition of a software system on the classes of objects the system manipulates and not on the functions the system performs. Each object encapsulates the data it manipulates and the functions, or methods, it performs.

Objects communicate by sending well-defined messages. Objects that have the same characteristics and behavior are grouped into classes. Further specialization is accomplished through subclassing, and a subclass inherits the behavior of its superclass. Additional behavior specific to the subclass can also be defined locally. The combination of encapsulation and inheritance supports the reuse of objects, while eliminating redundancy and allowing localization of any changes.

The object databases, that include and extend the semantic database notion, provide data management capabilities for the objects. These include persistence, transaction management, concurrency control, query capabilities, and so forth. Expert systems have traditionally combined object modeling with inferencing to build powerful systems to address problems in restricted domains (e.g., diagnosis). Less expensive hardware and higher bandwidth networks have made it possible to improve individual productivity and enhance group communication.

It is interesting to note that some innovative companies have already gained a competitive edge by adapting one or more of these technologies in their development environments. Object-oriented programming concepts and visual metaphors have already been thoroughly exploited by Apple, Inc. [4].
Bellcore claims significant productivity improvements using a semantic database [5]. TRW has reported improvements using workstation-based programming environments [6]. Currently, there are attempts to build various expert systems to address different aspects of the software life cycle [7].

Upon examining the recent trends, it is only natural for me as a researcher to speculate that the confluence of these technologies, as they evolve, will produce a new "object" data model that will:

- Merge the database and programming aspects discussed above
- Allow the seamless integration of constraint definition and inferencing at an object level
- Make reuse of objects easier with powerful browsers and intelligent objects to reason (using inferencing).

There are three areas in which this new object data model can play an important role in the next generation software environments. These are: rapid prototyping, software development, and software maintenance. In the following sections, we will speculate as to how these three areas could benefit from the evolving object paradigm.

(1) **Rapid prototyping platform.** Studies show that many complex systems are designed by a few superconceptualizers [8]. Often, these individuals are not programmers, but are people with the ability to synthesize and integrate knowledge from many domains. They frequently cast the genetic imprint of the software systems very early in the design process. Here, the powerful object data model, with its hierarchical class structure and inheritance, could allow these superconceptualizers to use design constructs that closely model the levels of abstraction often required in building complex systems.

The power of object modeling is clearly demonstrated by many applications in the AI and expert systems domain where such tools and environments are commonplace based on LISP. Unfortunately, this modeling capability has not made its way to traditional software engineering environments. However, recent developments in C-based languages incorporating the object models (C++ and objective-C) will increase the scope and power of similar tools and environments.

Additionally, the object databases in these environments will make reuse of objects easier with powerful browsers and intelligent query capabilities. Object-based user interface management systems will facilitate rapid prototyping. As the cumulative domain-specific knowledge accrues as an objectbase (which is a collection of these objects), it will be easier to change functionality (by subclassing) in the prototyping stage.

(2) **Software production environment.** The software life-cycle process is a candidate for modeling by the object paradigm [7]. Such a model imposes structure on different activities involved in the software process. In addition, an object database provides the data management involved in a software life-cycle. It contains the entities that are involved in the software production process. These include project history and status, various development tools, and knowledge about different objects and their relationships.

Such a powerful environment will allow automation and inter-connection between different phases of the software life-cycle that previously were not possible. It will also be possible to integrate different CASE tools in this environment in a transparent manner.

(3) **Software maintenance environment.** The object model also provides a basis to analyze and maintain existing programs written in C or other languages. Information necessary to maintain a program comes from different sources, but most often it is the source code itself.

Next generation software maintenance environments will be able to build objectbases from the source code and other relevant information. These objectbases will have knowledge about programming languages, inter-relationships of different program entities, and some knowledge about the software process itself. With such an environment, it will be possible to explore, analyze, and change the programs easily. It will also be possible to observe the ripple effects of any changes. Rules about the maintenance process itself will lend a certain degree of automation.

**Conclusion**

In this paper we have identified the confluence of several new technologies that will allow major improvements in the software development process. Future research in this area will encompass advances in database technology, programming languages, expert systems, and distributed workstations. This will provide powerful environments in the future to model, build, and maintain complex software systems with increased productivity.

**References**


