

Enterprise Architecture and Component Development

Alan Perkins
Visible Systems Corporation, USA

This presentation describes the benefits and techniques for using an enterprise information architecture to develop and manage reusable software components.

The benefits of software component development are often quoted as reusability, modularity, avoidance of duplication, reliability, and flexibility for expansion. Yet very few organizations have been successful in managing their development to realize these benefits. While components could and should be reusable, they usually are not. This is primarily because the components are developed independently with little or no regard for enterprise needs or standards.

The presentation will describe a rigorous and repeatable method that allows organizations to realize full potential of component development and further allows technological independence and tightly defined links between logical models, business requirements, and physical component designs.

The presentation will also describe automated tools that support the entire life cycle of component development and provide the means to document and link the various elements of an enterprise information architecture. These tools are the enabling technology that allow data sharing, software component reuse, and rapid change management to be realized.

Attendees will gain an appreciation for the value of using an enterprise information architecture for effective component development. They will also learn processes and critical success factors for successfully developing and managing truly reusable software components.

Bio

Alan Perkins has more than thirty years of business and information management experience. He specializes in helping enterprises manage and meet strategic goals through the use of effective, proven management methods like Enterprise Engineering, Business Process Reengineering, Total Quality Management, and Information Engineering. He focuses on the use of information technologies such as executive information systems and data warehousing to support enterprise management processes. His approach results in structure, standards, and a knowledge base which facilitates enterprise-wide communication and performance measurement.

Mr. Perkins has written several papers for Visible -- his most recent papers describe the Visible Enterprise Engineering methodology, the Visible approach to developing strategic information warehouses, the real solution to the Y2K problem, Software Component Development, and how Visible can help government organizations implement the Government Performance and Results Act (GPRA).

In the past few years, Mr. Perkins has been a featured speaker at several conferences, including the 1998 Conference of the Special Interest Group on Computer Personnel Research, the annual Southwest Government Technology Conference, Institute of Industrial Engineering (IIE) Symposia on Productivity and Quality Improvement with a Focus on Government, the annual PowerSoft conference, and many national conferences sponsored by the International Quality and Productivity Center.

Enterprise Architecture and Object-Oriented Development

Alan Perkins
Vice President, Consulting Services
Visible Systems Corporation
Alan.Perkins@Visible.com

Abstract.

This paper describes the benefits of using an enterprise information architecture to develop and manage reusable software components -- specifically objects.

The paper also describes Enterprise Engineering -- a rigorous and repeatable methodology that allows organizations to realize the full potential of component development. Enterprise Engineering features technology independence and tightly defined links between business requirements, logical component models, and physical component designs.

1. Introduction.

Object-oriented (OO) analysis, design, and development techniques have become increasingly popular as a means to better information systems in support of complex enterprises.

At the heart of OO is the "object," which combines data structure and behavior (process) in a single component. The inner workings of an object are usually hidden; only its visible interfaces need to be understood by any other processes (or objects) that interact with it.

This is in contrast to conventional programming, in which data structures and their behaviors are defined separately and only loosely connected [Rumbaugh, 1991, p].

The following are more precise definitions of OO analysis and design [Booch, 1994]:

OO analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

OO design is a method of design encompassing the process of OO decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the systems under design.

OO analysis and design is one of many means of developing reusable, shareable software components. The benefits of an OO approach to software component development are reusability, modularity, avoidance of duplication, reliability, and flexibility for expansion -- yet very few organizations **have been** successful in managing object development to realize these benefits. While objects could and should be reusable and shared, they usually are not. This is primarily because objects are easy to develop, but extremely difficult to manage.

2. The Problems With OO.

Software developers have been building computer applications from reusable components for years. In the early 1960's, when computer programs were developed and executed using punched paper cards, it made sense to reuse modules and subroutines in as many applications as possible. The components had already been tested and nobody wanted to redesign and re-key-punch routines that worked. In large organizations, there were filing cabinets full of programs and subroutines (components) that were routinely reused and shared by all developers.

With the advent of digital storage for source code (copy libraries, etc.) and the ability to modify the code

using automated tools, developers stopped sharing their components and often didn't even reuse them. Most recently, graphical OO development tools have made it even easier to proliferate redundant and duplicative software components.

Most organizations that have been adopting OO techniques and tools, have not been managing their use. True software component reuse and sharing requires proper management of component development. This means establishing and enforcing standards. It also means using standard techniques and best practices

Establishing libraries of object classes is one common practice that many organizations use as the basis for object reusability. Objects with the same data structure and similar behavior are grouped into an object "class." Object libraries are either built from objects designed during internal projects or purchased as "off-the-shelf" commercial products. However, there are practical problems facing organizations that establish and use such libraries:

- Reuse enforces standards when new objects are created, which is likely to result in additional time and cost compared with generating a "single-use" object.
- The performance pressures placed on those responsible for systems implementation usually focuses on fast delivery of new or additional functionality to users, rather than the more long-term benefits of object reuse and sharing.
- Locating the right object classes or code components in available libraries is not easy.
- There is a danger that reusable code components can become too large and generic to be effective or **efficient**.

In order to overcome these problems and achieve the benefits of objects (rapid development, reuse, easier maintenance, and extensibility), OO analysis and design must be part of a rigorous and repeatable methodology.

3.00 and Enterprise Engineering.

Enterprise Engineering (EE) is a unified methodology that uses an Enterprise Architecture as the framework for developing and managing reusable software objects.

Object-Oriented and Enterprise Engineering are symbiotic. The logical data and process models that comprise an Enterprise Information Architecture are

technology independent and reflect the business goals and critical success factors of the enterprise. They can be revised, amended, and refined independently from the implementation environment of physical objects. The logical data and process models provide a firm base from which OO designs are developed. Linking objects to business requirements through an Enterprise Architecture Framework ensures that objects are developed only to meet specific business needs. Using an architecture also facilitates object management and allows organizations to realize the full potential of OO techniques and tools.

Architecture models also bridge the gap between business information requirements and information system objects and allow improved communication between business experts (information users) and system designers (information system providers).

OO analysis and design represent the "physical" phases of the Enterprise Engineering life cycle. The EE life cycle is illustrated in Figure 1.

4. Engineering Quality Information Systems.

Quality information systems are more than just error-free code. They also have the following characteristics:

- Support Enterprise Strategic Objectives
- Meet Business Area Requirements
- Are Reliable, Flexible, and Scaleable
- Share Corporate Data and Standards
- Built from Reusable Components
- Delivered On Time and On Budget

EE is the unified method that allows consistent quality information system development from software components.

First, business requirements are defined in real-world terms: goals, strategies, objectives, tasks, critical success factors, performance metrics, etc.

As the business requirements are being defined, an enterprise information architecture is developed using logical models that are both unambiguous to developers and understandable to business experts.. This narrows the typical communications gap between system users and those who will implement and maintain the systems

The enterprise information architecture is made up of logical data and activity models that represent the information and business rules necessary to support the defined business requirements.

