Architecting for Large-Scale Systematic Component Reuse

Martin L. Griss
Hewlett-Packard Company, Laboratories
1501 Page Mill Road, 1U-16
Palo Alto, CA 94304-1126, USA
+1-650-857-8715
griss@hpl.hp.com

ABSTRACT
Organizations building highly complex business and technical systems must architect families of systems and implement these with large-scale component reuse. Without carefully architecting the systems, components, organizations and processes for reuse, reuse will not succeed. The talk explains our experience with component-based software reuse and a systematic reuse process using UML, and object-oriented business and system modeling

Keywords
Reuse, architecture, components, process, organization

1 SOFTWARE REUSE IS STRATEGIC

To meet competitive pressures and provide increased value to customers, many organizations are radically reengineering their business processes. Timely and cost-effective creation of new mission-critical software systems, and dramatically improved software development is crucial. Strategic software reuse promises tremendous improvements in time to market, quality and cost. "Silver bullets," such as CASE, OO, RAD or Java, will not solve the software problem. While OO and component technology are significant, successful reuse programs must address many other business, organization, process, architectural, cultural and technology issues.

Experience at HP, Ericsson, Rational and several other companies, led to our holistic approach to architected reuse - the *Reuse-driven Software Engineering Business* (RSEB)[1]. We systematically move from the business processes of an enterprise, through the system architecture for a family of applications that support these business processes, to the design and use of highly reusable component systems.

2 BUSINESS NEED MOTIVATES REUSE

The change to a reuse-oriented architecture, process and organization requires a significant investment and effort. Architecture, reuse, standards, process, organization and business issues are closely related, and must be addressed within a common framework, following an incremental transition plan[1]. A successful, large-scale architected reuse effort must be:

Business-driven - The organization has an articulated and compelling need for dramatic improvements in cycle time, cost, productivity and/or agility. They produce software applications, embedded systems or subsystems that form an obvious product-line, application family or "domain."

Architected - Applications, systems and components are designed to fit together, and to cover the needs of the family or domain. A layered, modular structure, and standards shape the use of OO technology.

Process-oriented - Distinct reuse-oriented software development and maintenance processes for architecture, components and applications are followed. They identify and express commonality and variability, and design and package components for reuse.

Organized - Long-term management leadership and commitment ensures that the organization is structured, trained and staffed to follow the processes and standards. Several distinct subgroups create, reuse and support reusable components.

A common engineering notation to describe architectures, systems, processes and organizations is key. This notation and a compendium of common, reusable architectures, designs and patterns provide a base for more reusable components and frameworks. We use the Unified Modeling Language (UML) [6], with extensions to model layered architectures, variability, reusable and customizable components, interfaces and frameworks.

Reuse techniques are integrated into the Object-Oriented Software Engineering process (OOSE)[5], to produce an incremental/iterative model-driven reuse development process[2,3,4]. We also use OO Business Engineering[7] to

model business processes, business objects, reuse processes, organization designs and BPR-driven transition. Business models precisely define the details of the processes, how they interact, and the roles of various actors and workers.

3 ARCHITECTURE, PROCESS AND ORGANIZATION FOR APPLICATION FAMILIES

An RSEB is a software development organization optimized to employ reuse-based software engineering and management techniques. Often, a dominant and compelling business goal is to reduce product development times, yet retain market agility within and across product families. Examples of these families are applications and systems built to meet different customer and country needs by combining and customizing reusable components: measurement systems built from common instrument components, and Telecom switches (such as Ericsson' AXE) which are configured to a variety of situations.

A strategic decision must be made to establish one or more reuse-driven business units within these organizations. As a reuse-driven software organization, such business units produce multiple, related applications, centered and optimized around the production and reuse of components. These applications commonly form a product-line or product family. As a software business, technical tradeoffs must be managed using well defined economic, product and process measures.

Each application or component is expressed as a connected set (a system) of OO models and software using OOSE and UML notation. Components are public elements from the various models, including usecases, analysis types, interfaces and design and implementation classes (code). Components are not used alone, but in groups called component systems and frameworks. Component systems are constructed by a Component System Engineering process. Application systems are constructed in an Application System Engineering process.

Shaping the applications, and ensuring high levels of reuse is the architecture of the reusable components and applications. Applications and components interact with each other in a layered, modular architecture; patterns and frameworks ensure consistent structure and mechanisms. An Application Family Engineering process develops the layered architecture. A top application layer consists of related (a family of) application systems supporting a business, extensively reusing underlying component systems. Next are cross-application components reusable only for the specific business or application domain area, such as banking systems or microwave instruments. Below are cross-business middleware or platform specific components including graphical user interfaces, ORB's, OS, HW drivers, etc.

4 INCREMENTAL TRANSITION

With appropriate management commitment, such as a strategic statement from the CEO to improve the speed with which new products are developed, change begins. The transition of an existing software organization into an RSEB follows an OO Business Engineering framework[7]. We express reuse-oriented process and organizations using UML business models. An appropriate software reuse organization is defined and installed, matching key existing processes and organization process maturity. Well-known social and cultural impediments to reuse are addressed. RSEB transition steps motivate, prepare for and drive the change, including process modeling, organization reverse engineering, change management techniques , champions, incremental pilots, and distinct reuse-maturity stages[2,3].

5 CONCLUSION

The RSEB approach provides a systematic framework for the definition and transition to a reuse-driven organization. Clear business goals and management commitment are key to making the process and organization changes needed for architected reuse. UML as a common modeling language for OO system and business modeling, provides a widely accepted software "blueprinting language" for architects and engineers to describe complex systems and precisely define their reusable components.

REFERENCES

1 I Jacobson, M Griss, P Jonsson, Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley-Longman, May 1997.

2 ML Griss, Software reuse - a process of getting organized, Object Magazine, May 1995. (See http://www.hpl.hp.com/reuse for other columns and references.)

3 ML Griss, Systematic OO software reuse - a year of progress, Object Magazine, February 1996.

4 I Jacobson, Succeeding with Objects: Reuse in Reality, Object Magazine, July 1996.

5 I Jacobson et al, Object-Oriented Software Engineering: A Usecase-driven Approach, Addison-Wesley, 1992.

6 G Booch, J Rumbaugh, and I Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.

7 I Jacobson, M Ericsson and A Jacobson, The Object Advantage: Business Process Reengineering with Object Technology, Addison-Wesley 1994.