Object technology is believed to be crucial in achieving the long sought-after goal of widespread reuse. This goal is the most frequently stated reason for adopting OT. Unfortunately, many people naively equate reuse with objects, expecting it to “automatically” ensure reuse, but often do not get much reuse. Based on my experience with reuse at HP, Objectory and Rational, and with many customers, I know that without extensive changes to support component-based development and systematic reuse, OT as used today will not succeed in giving users reuse. Without an explicit reuse agenda, and a systematic approach to the design and use of reusable components and frameworks, OT reuse will not succeed. In almost all cases of successful reuse, architecture, a dedicated component development and support group, management support, and a stable domain were the keys to success. These largely non-technical issues seem to be more important to successful reuse than the specific language or design chosen.

In this tutorial I address this architecture/process/organization question systematically. I describe an exciting and coherent approach, which applies Object-Oriented Business Engineering concepts of BPR to restructure a software development organization to offer reuse at a large scale. This is outlined as a generic and customizable “Reuse Business” framework. Since the transition to a Reuse Business is a complex and risky endeavor, it is important to do it systematically. The approach therefore combines the techniques of BPR, change management and incremental reuse adoption.

To obtain a true systematic OT reuse process, I have augmented the Use Case Driven Object-Oriented Software Engineering (OOSE) with specific reuse-oriented architectures and processes, coupled with organization and infrastructure guidelines and training. The work is based on the new Unified Modeling Language (UML 1.1), taking full advantage of the features for packages, stereotypes, facades, patterns, etc.

I have extended the OOSE process to start from a business model of the target organization, with specific architecture and component design steps. Component and framework engineering includes explicit design and implementation guidelines, building on a catalog of architectures, patterns, designs and mechanisms. I provide a structured approach to produce layered, modular architectures.

Bio

Martin L. Griss is a Principal Laboratory Scientist at Hewlett-Packard Laboratories, Alto, California, where for the last 15 years he has researched software engineering processes and systems, systematic software reuse, and object-oriented development. He created and led the first HP corporate reuse program. He led HP efforts to standardize UML for the OMG. He was previously director of the Software Technology Laboratory at HP Laboratories, and an Associate Professor of Computer Science at the University of Utah. He has more than 25 years of experience in software engineering, is co-author of the best-selling book *Software Reuse: Architecture, Process and Organization for Business Success*, writes a column for *Object Magazine*, has written more than 40 articles and lectures widely on systematic reuse and software process improvement. He received a Ph.D. (Physics) from the University of Illinois in 1971.

The Rational Unified Process is a process product, developed and maintained by Rational® Software. The development team for the Rational Unified Process are working closely with customers, partners, Rational's product groups as well as Rational's consultant organization, to ensure that the process is continuously updated and improved upon to reflect recent experiences and evolving and proven best practices. The Rational Unified Process (RUP) is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003. RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs. RUP is a specific implementation of the Unified Process. Software Architecture: Process and Method, Representation, Techniques, Tools, and Business Implications. It provides a good introduction to several SEI architectural methods. Software Reuse: Architecture, Process and Organization for Business Success. Addison-Wesley, 1997. As the title indicates, this book. The role of SA is essential as never before, so much so that no organization undertakes 'serious' projects without the support of suitable architecture practices.