Chapter 2
The Rational Unified Process

THIS CHAPTER GIVES an overview of the Rational Unified Process, introduces the process structure, describes the process product, and outlines its main features.

WHAT IS THE RATIONAL UNIFIED PROCESS?

The Rational Unified Process is a *software engineering process*. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

The Rational Unified Process is a *process product*. It is developed and maintained by Rational Software and integrated with its suite of software development tools. It is available from IBM on CD-ROM or through the Internet. This book is an integral part of the Rational Unified Process but represents only a small fraction of the Rational Unified Process knowledge base. Later in this chapter we describe the physical structure of the process product.

The Rational Unified Process is also a *process framework* that can be adapted and extended to suit the needs of an adopting organization. Chapter 3 describes in more detail how the process framework is organized and introduces the *process model*, the elements that compose the process framework.

The Rational Unified Process captures many of the *best practices* in modern software development in a form that is suitable for a wide range of projects and organizations. Along with many others, it covers the practices introduced in Chapter 1:

1. Develop software iteratively.
2. Manage requirements.
3. Use component-based architectures.
4. Visually model software.
5. Continuously verify software quality.
6. Control changes to software.

THE RATIONAL UNIFIED PROCESS AS A PRODUCT

Many organizations have slowly become aware of the importance of a well-defined and well-documented software development process to the success of their software projects. Over the years, they have collected their knowledge and shared it with their developers. This collective know-how often grows out of methods, published textbooks, training programs, and small how-to notes amassed over several projects. Unfortunately, these practices often end up gathering dust in nice binders on a developer's shelf—rarely updated, rapidly becoming obsolete, and almost never followed.

"Software processes are software, too," wrote Lee Osterweil.[1] In contrast with the dusty-binder approach, the Rational Unified Process is designed, developed, delivered, and maintained like any software tool. The Rational Unified Process shares many characteristics with software products:

- IBM releases regular upgrades.
- It is delivered online using Web technology, so it is literally at the fingertips of the developers.

1. Lee Osterweil, "Software Processes Are Software Too," *Proceedings of the Ninth International Conference on Software Engineering*, pp. 2–13, Mar. 30–Apr. 2, 1987, Monterey, CA.

- It can be tailored and configured to suit the specific needs of a development organization.
- It is integrated with many of the software development tools in the IBM Rational Suites so that developers can access process guidance from within the tool they are using.

This approach of treating the process as a software product provides the following benefits:

- The process is never obsolete; at regular intervals companies get new releases with improvements and up-to-date techniques.
- All project members can access the latest version of their process configuration on an intranet.
- Java applets, such as a process browser and a built-in search engine, allow developers to reach instantaneously process guidance or policies, including the latest document templates they should use.
- Hyperlinks provide navigation from one part of the process to another, eventually branching out to a software development tool or to an external reference or guideline document.
- Local, project- or company-specific process improvements or special procedures are included easily.
- Each project or department can manage its own version or variant of the process.

This online Rational Unified Process product gives you benefits that are difficult to achieve with a process that is available only in the form of a book or binder.

Organization of the Process Product

The RUP product is delivered on a CD-ROM or via the Internet and consists of the following:

1. An online version of the Rational Unified Process, which is the electronic guide for the RUP: a fully hyperlinked Web site description of the process in HTML

2. A tool called RUP Builder that allows you to compose and publish your own tailored configuration of the Rational Unified Process (see Chapter 14)
3. Additions to the Rational Unified Process, called Process Plug-ins, which cover specialized topics and which can be added to your configuration with RUP Builder. Many such plug-ins are available for download from the Rational Developers Network or can be obtained from IBM partners

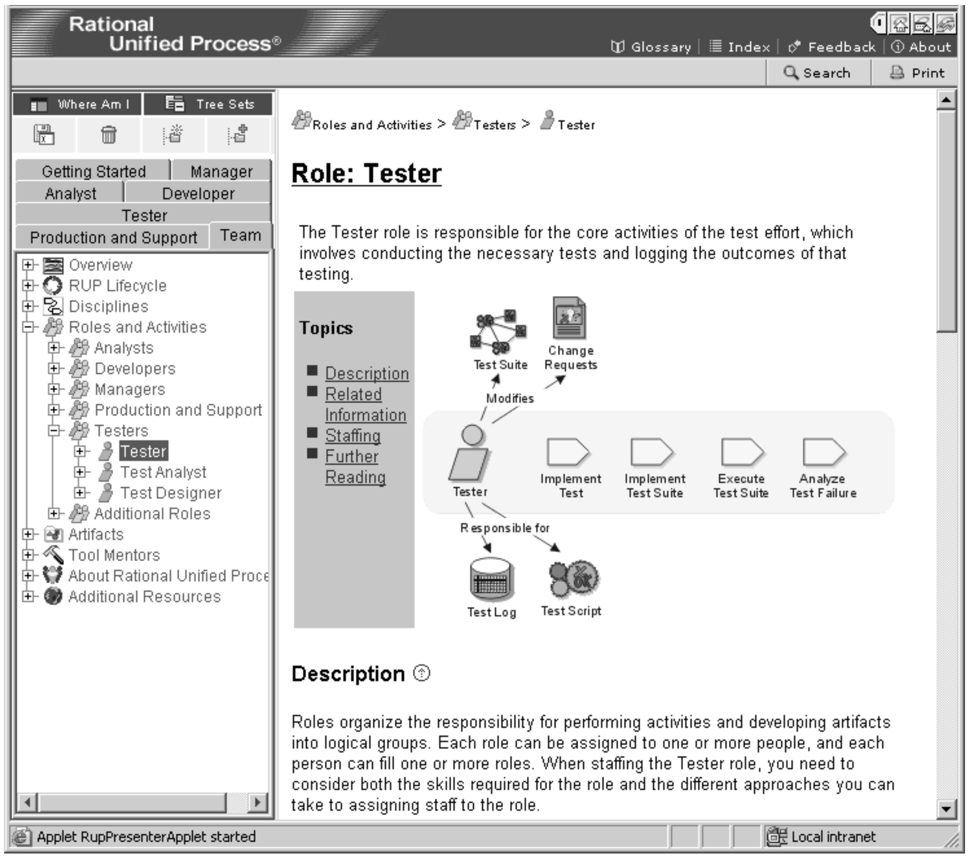
A RUP configuration can be used with any of the popular Web browsers, such as Netscape Navigator™ and Microsoft Internet Explorer™. Information is easy to find, thanks to these elements:

- Extensive hyperlinking
- Graphical navigation (Most graphical elements are hyperlinked to the process elements they depict.)
- A hierarchical tree browser
- An exhaustive index
- A built-in search engine

A tool called MyRUP enables navigation through a RUP configuration and allows a user to define a personalized view of a RUP configuration. You can locate these facilities as shown in Figure 2-1, which is a snapshot of a RUP page.

In a process configuration, you will find not only a complete description of the process itself but also the following:

- Tool mentors, which provide additional guidance when you're working with any of the software development tools offered by IBM Rational Software, such as Rational XDE (eXtended Develoment Environment) for visual modeling and Rational ClearCase for configuration management
- Templates for all major process artifacts, for example:
  - Microsoft Word templates and Adobe FrameMaker templates for most plain documents and reports
  - Rational SoDA templates, to automate the assembly of documents from multiple sources
  - RequisitePro templates to help manage requirements



Role: Tester

The Tester role is responsible for the core activities of the test effort, which involves conducting the necessary tests and logging the outcomes of that testing.

Description

Roles organize the responsibility for performing activities and developing artifacts into logical groups. Each role can be assigned to one or more people, and each person can fill one or more roles. When staffing the Tester role, you need to consider both the skills required for the role and the different approaches you can take to assigning staff to the role.

FIGURE 2-1 ***The Rational Unified Process online***

- – Microsoft Project templates, to help plan an iterative project based on the RUP
- – HTML templates for extending the online process itself

- Examples of artifacts for simple projects

For the Process Engineers

For process engineers additional tools and guidance are available. The Rational Process Workbench (RPW) is the tool used to define process components, perform extensive modification to RUP, and create process plug-ins. It is composed of the RUP Modeler, an add-in to Rational XDE, and RUP organizer. A separate process configuration, called the Process Engineering Process, provides guidance on the tailoring, the customization of RUP, the creation of plug-ins, and the deployment of the Rational Unified Process in an organization. We will examine these topics in Chapters 14 and 17.

Process Structure: Two Dimensions

Figure 2-2 shows the overall *architecture* of the Rational Unified Process. The process has two structures or, if you prefer, two dimensions:

1. The horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds.
2. The vertical axis represents core process disciplines, which group activities logically by nature.

The first dimension represents the dynamic aspect of the process as it is enacted, and it is expressed in terms of cycles, phases,

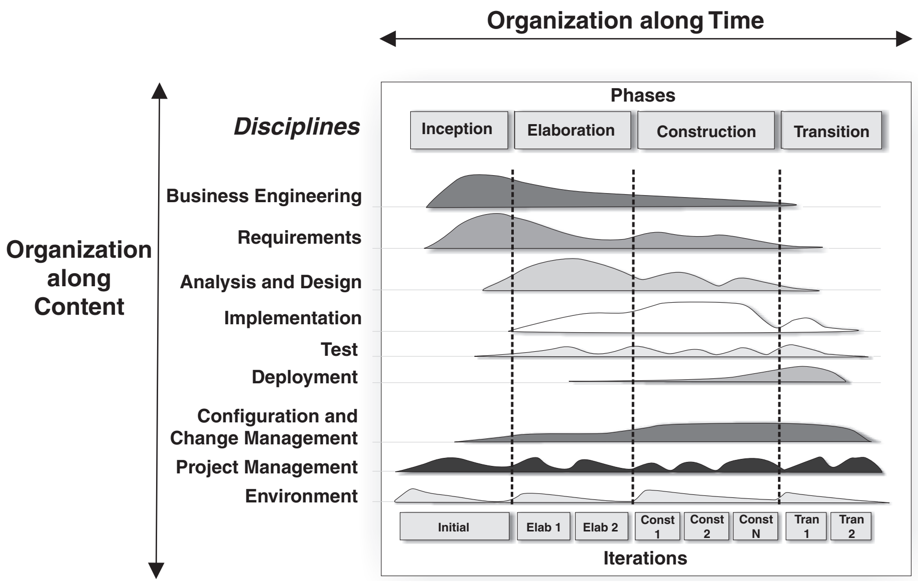


FIGURE 2-2 ***Process structure—two dimensions***

iterations, and milestones. This is described further in Chapter 4, Dynamic Structure: Iterative Development, and in Chapter 7, The Project Management Discipline.

The second dimension represents the static aspect of the process: its description in terms of process components, activities, disciplines, artifacts, and roles. This is described further in Chapter 3, Static Structure: Process Description.

SOFTWARE BEST PRACTICES IN THE RATIONAL UNIFIED PROCESS

This section revisits the best practices introduced by Grady Booch in Chapter 1 and maps them to major components of the Rational Unified Process. The RUP, however, contains many other best practices in all aspects of software development.

Iterative Development

The iterative approach recommended by the Rational Unified Process is generally superior to a linear, or waterfall, approach for a number of reasons:

- It lets you take into account changing requirements. The truth is that requirements usually change. Changing requirements and requirements "creep" have always been primary sources of project trouble, which lead to late delivery, missed schedules, unsatisfied customers, and frustrated developers.
- In the Rational Unified Process, integration is not one "big bang" at the end; instead, elements are integrated progressively. This iterative approach is almost a process of continuous integration. What used to be a lengthy time of uncertainty and pain—taking up to 40% of the effort at the end of a project—is now broken into six to nine smaller integrations that begin with far fewer elements to integrate.
- The iterative approach lets you mitigate risks earlier because integration is generally the only time that risks are discovered or addressed. As you unroll the early iterations,

you go through all process components, exercising many aspects of the project, such as tools, off-the-shelf software, people skills, and so on. Perceived risks will prove not to be risks, and new, unsuspected risks will be revealed.

- It provides management with a means of making tactical changes to the product for whatever reason, for example, to compete with existing products. You can decide to release a product early with reduced functionality to counter a move by a competitor, or you can adopt another vendor for a given technology.
- It facilitates reuse because it is easy to identify common parts as they are partially designed or implemented instead of identifying all commonality in the beginning before anything has been designed or implemented. Identifying and developing reusable parts is difficult. Design reviews in early iterations allow architects to identify unsuspected potential reuse and then develop and mature common code for it in subsequent iterations.
- It results in a very robust architecture because you correct errors over several iterations. Flaws are detected even in the early iterations as the product moves beyond inception into elaboration rather than in one massive testing phase at the end. Performance bottlenecks are discovered at a time when they can still be addressed instead of creating panic on the eve of delivery.
- Developers can learn along the way, and their various abilities and specialties are employed more fully during the entire lifecycle. Testers start testing early, technical writers write early, and so on. In a noniterative development, the same people would be waiting around to begin their work, making plan after plan but not making concrete progress. What can a tester test when the product consists only of three feet of design documentation on a shelf? Training needs or the need for additional people is spotted early, during assessment reviews.
- The development process itself can be improved and refined along the way. The assessment at the end of an iteration not

only looks at the status of the project from a product/schedule perspective but also analyzes what should be changed in the organization and in the process to make it perform better in the next iteration.

Project managers often resist the iterative approach, seeing it as a kind of endless and uncontrolled hacking. In the Rational Unified Process, the iterative approach is very controlled; iterations are planned in number, duration, and objectives. The tasks and responsibilities of the participants are defined. Objective measures of progress are captured. Some reworking takes place from one iteration to the next, but this, too, is controlled carefully.

Chapter 4 describes this iterative approach in more detail, and Chapter 7 describes how to manage an iterative process and, in particular, how to plan it.

Requirements Management

Requirements management is a systematic approach to eliciting, organizing, communicating, and managing the changing requirements of a software-intensive system or application.

The benefits of effective requirements management include the following:

- *Better control of complex projects*
  Lack of understanding of the intended system behavior and requirements "creep" are common factors in out-of-control projects.
- *Improved software quality and customer satisfaction*
  The fundamental measure of quality is whether a system does what it is supposed to do. This can be assessed only when all *stakeholders*[2] have a common understanding of what must be built and tested.

2. A stakeholder is any person or representative of an organization who has a stake—a vested interest—in the outcome of a project. A stakeholder can be an end user, a purchaser, a contractor, a developer, a project manager, or anyone else who cares enough or whose needs must be met by the project.

- *Reduced project costs and delays*
  Fixing errors in requirements is very expensive; therefore, decreasing these errors early in the development cycle cuts project costs and prevents delays.
- *Improved team communication*
  Requirements management facilitates the involvement of users early in the process, helping to ensure that the application meets their needs. Well-managed requirements build a common understanding of the project needs and commitments among the stakeholders: users, customers, management, designers, and testers.

In Chapter 9, The Requirements Discipline, we revisit and expand on this important feature of the Rational Unified Process. Chapter 13, The Configuration and Change Management Discipline, discusses the aspects related to tracking changes.

Architecture and Use of Components

Use cases drive the Rational Unified Process throughout the entire lifecycle, but the design activities are centered on the notion of *architecture*—either system architecture or, for software-intensive systems, software architecture. The main focus of the early iterations of the process is to produce and validate a software architecture that, in the initial development cycle, takes the form of an executable architectural prototype that gradually evolves to become the final system in later iterations.

The Rational Unified Process provides a methodical, systematic way to design, develop, and validate an architecture. It offers templates for describing an architecture based on the concept of multiple architectural views. It provides for the capture of architectural style, design rules, and constraints. The design process component contains specific activities aimed at identifying architectural constraints and architecturally significant elements, as well as guidelines on how to make architectural choices. The management process shows how planning the early iterations takes into account the design of an architecture and the resolution of the major technical risks.

A software *component* can be defined as a nontrivial piece of software, a module, a package, or a subsystem that fulfills a clear function, has a clear boundary, and can be integrated into a well-defined architecture. It is the physical realization of an abstraction in your design. Component-based development can take various flavors:

- In defining a modular architecture, you identify, isolate, design, develop, and test well-formed components. These components can be tested individually and integrated gradually to form the whole system.
- Furthermore, some of these components can be developed to be reusable, especially the components that provide common solutions to a wide range of common problems. These reusable components are typically larger than mere collections of utilities or class libraries. They form the basis of reuse within an organization, thereby increasing overall software productivity and quality.
- More recently, the advent of commercially successful infrastructures that support the concept of software components has launched a whole industry of off-the-shelf components for various domains, allowing developers to buy and integrate components rather than develop them in-house.

The first point exploits the old concepts of modularity and encapsulation, bringing the concepts underlying object-oriented technology a step further. The final two points shift software development from programming software (one line at a time) to composing software (by assembling components).

The Rational Unified Process supports component-based development in several ways:

- The iterative approach allows developers to identify components progressively and decide which ones to develop, which ones to reuse, and which ones to buy.
- The focus on software architecture allows you to articulate the structure. The architecture enumerates the components and the ways they integrate as well as the fundamental mechanisms and patterns by which they interact.

- Concepts such as packages, subsystems, and layers are used during analysis and design to organize components and specify interfaces.
- Testing is organized around single components first and then is gradually expanded to include larger sets of integrated components.

Chapter 5 defines and expands the concept of architecture and its central role in the Rational Unified Process.

Modeling and the UML

A large part of the Rational Unified Process is about developing and maintaining *models* of the system under development. Models help us to understand and shape both the problem and its solution. A model is a simplification of the reality that helps us master a large, complex system that cannot be comprehended in its entirety. We introduce several models in this book: a use-case model (Chapter 6), business models (Chapter 8), and design models and analysis models (Chapter 10).

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives you a standard means of writing the system's blueprints, covering conceptual items such as business processes and system functions as well as concrete items such as classes written in a specific programming language, database schemas, and reusable software components.[3]

The UML is a common language to express the various models, but it does not tell you how to develop software. It provides the vocabulary, but it doesn't tell you how to write the book. That is why Rational has developed the Rational Unified Process hand-in-hand with the UML to complement our work with the UML. The Rational Unified Process is a guide to the effective use of the UML for modeling. It describes which models you need, why you need them, and how to construct them. RUP 2000 and later versions use UML version 1.4.

3. Grady Booch et al., *UML Users Guide.* Reading, MA: Addison-Wesley, 1998.

Configuration and Change Management

Particularly in an iterative development, many work products are modified and modified often. By allowing flexibility in the planning and execution of the development and by allowing the requirements to evolve, iterative development emphasizes the vital issues of tracking changes and ensuring that everything and everyone is in sync. Focused closely on the needs of the development organization, change management is a systematic approach to managing changes in requirements, design, and implementation. It also covers the important activities of keeping track of defects, misunderstandings, and project commitments as well as associating these activities with specific artifacts and releases. Change management is tied to configuration management and to measurements.

Chapter 13, The Configuration and Change Management Discipline, expands on these important aspects of software management and their interrelationships.

OTHER KEY FEATURES OF THE RATIONAL UNIFIED PROCESS

In addition to these best practices, three important features of the Rational Unified Process are worth mentioning now:

- The role of use cases in driving many aspects of the development
- Its use as a process framework that can be tailored and extended by an adopting organization
- The need for software development tools to support the process

Use-Case-Driven Development

It is often difficult to look at a traditional object-oriented system model and tell how the system does what it is supposed to do. We believe that this difficulty stems from the lack of a consistent, visible thread through the system when it performs certain tasks. In the Rational Unified Process, use cases provide that thread by defining the behavior performed by a system.

Use cases are not required in object orientation, but they provide important links between system requirements and other development artifacts such as design and tests. Other object-oriented methods provide use-case-like representation but use different names for it, such as *scenarios* and *threads*.

The Rational Unified Process is a *use-case-driven* approach, which means that the use cases defined for the system are the foundation for the rest of development process. Use cases play a major role in several of the process disciplines, especially requirements, design, test, and management. Use cases are also key to business modeling.

If you are unfamiliar with the concept of use case, Chapter 6, A Use-Case-Driven Process, introduces it in more detail and shows how use cases are used.

Process Configuration

You could envisage to use one of the predefined configurations of the RUP “as is.” But in doing so you run into the risk of doing too much, of getting lost in the details, or of developing artifacts that bring no value to your project in your specific context.

A process should not be followed blindly, generating useless work and producing artifacts that are of little added value. Instead, the process must be made *as lean as possible* while fulfilling its mission to rapidly produce predictably high-quality software. The adopting organization should complement the process with its own best practices and with its specific rules and procedures.

The Rational Unified Process is a process framework that the adopting organization can modify, adjust, and expand to accommodate the specific needs, characteristics, constraints, and history of its organization, culture, and domain. RUP 2003 offers two process *bases*—one for small projects and one for larger and more formal projects—and it brings an array of process components to choose from, to cover various additional disciplines (real-time system design, system engineering, user interface design), various technologies (IBM Websphere, Microsoft .NET, J2EE), programming languages, tools, and techniques.

Chapter 17, Implementing the Rational Unified Process, explains how the process is implemented in an adopting organization.

Tools Support

To be effective, a process must be supported by adequate tools. The Rational Unified Process is supported by a vast palette of tools that automate steps in many activities. These tools are used to create and maintain the various artifacts—models in particular—of the software engineering process, namely, visual modeling, programming, and testing. The tools are invaluable in supporting the bookkeeping associated with the change management and the configuration management that accompany each iteration.

Chapter 3 introduces the concept of *tool mentors,* which provide tool-related guidance. As we go through the various process disciplines in Chapters 7 through 15, we introduce the tools of choice to support the activities of each discipline.

Who Is Using the Rational Unified Process?

Probably about half a million users worldwide working in more than three thousand companies were using the Rational Unified Process in 2003. They were using it in various domains of applications and for large and small projects. This shows the versatility and wide applicability of the Rational Unified Process. Here are some examples of various uses in companies that have offices around the world:

- Telecommunications: Ericsson, Alcatel
- Transportation, aerospace, defense: Lockheed-Martin, British Aerospace
- Manufacturing: Xerox, Volvo, Intel
- Finances: Visa, Merrill Lynch, Schwab
- System integrators: CAP Gemini Ernst & Young, Oracle, Deloitte & Touche

The way these organizations use the Rational Unified Process varies greatly. Some use it very formally; they have evolved their own company process from the Rational Unified Process, which they follow with great care. Others use it more informally, a sort of electronic coach on software engineering, taking the Rational Unified Process as a repository of advice, templates, and guidance, which they use as they go along.

A BRIEF HISTORY OF THE RATIONAL UNIFIED PROCESS

The Rational Unified Process has matured over the years and reflects the collective experience of the many people and companies that today make up the rich heritage of IBM's Rational Software division. Let us take a quick look at the rich ancestry of RUP 2003, as illustrated in Figure 2-3.

Going backward in time, the Rational Unified Process was brought into the IBM offering by the acquisition of the 20-year-old Rational Software Corporation by IBM Software Group in February 2003. The Rational Unified Process incorporates material in the areas of data engineering, business modeling, project management, and configuration management, the latter as a result of a merger with Pure-Atria in 1997. It includes elements of the Real-Time Object-Oriented Method, developed by the founders of ObjecTime, acquired by Rational in 2000. It also brings a tighter integration to the Rational Software suite of tools.

The Rational Unified Process is the direct successor to the Rational Objectory Process (ROP), version 4, which was the result of the integration of the Rational Approach and the Objectory Process (version 3.8) after the merger of Rational Software Corporation and Objectory AB in 1995. From its Objectory ancestry, the process has inherited its process model (described in Chapter 3) and the central concept of use case (described in Chapter 6). From its Rational background, it gained the current formulation of iterative development (described in Chapters 4 and 11) and architecture (described in Chapter 5). This ROP version 4 also incorporated material on requirements management from Requisite, Inc., and a detailed test process inherited from SQA, Inc., companies that also were acquired by Rational Software. Finally, this version of the process was the first to use the newly created Unified Modeling Language (UML 0.8).

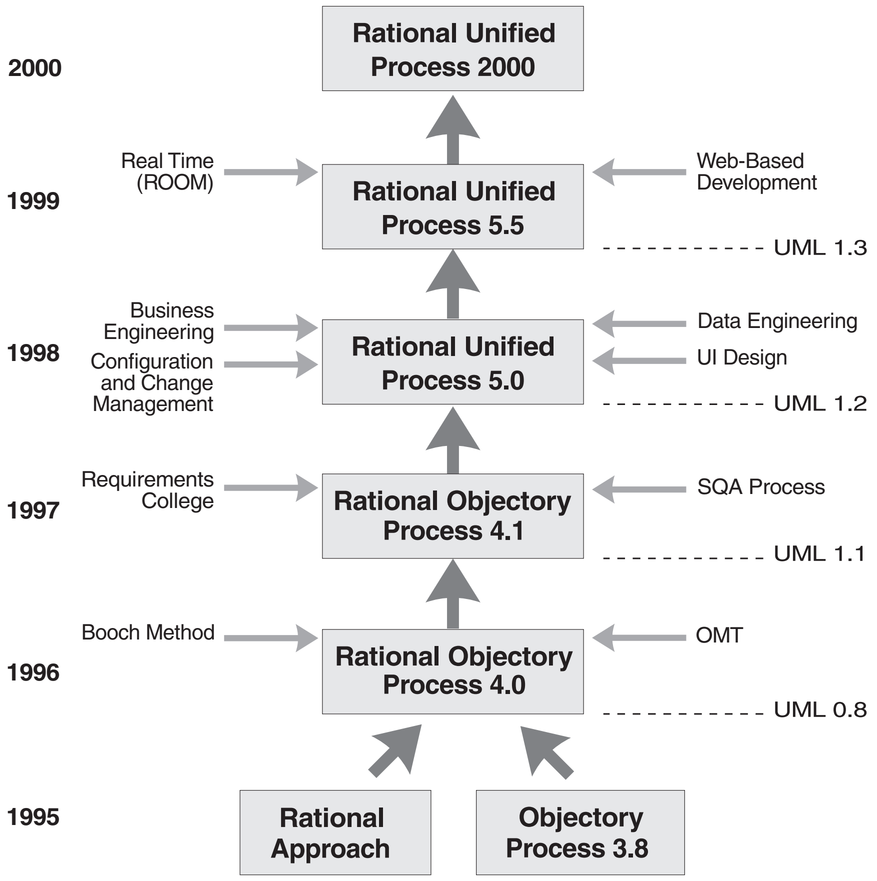


FIGURE 2-3 ***Genealogy of the Rational Unified Process***

The Objectory Process was created in Sweden in 1987 by Ivar Jacobson.[4]

SUMMARY

- The Rational Unified Process is a software development process that covers the entire software development lifecycle.

4. Ivar Jacobson et al., *Object-Oriented Software Engineering: A Use-Case-Driven Approach*. Reading, MA: Addison-Wesley, 1992.

- It is a process product that brings a wealth of knowledge, accumulated over 20 years from a vast range of sources, right to the practitioner's workstation in the form of an online electronic guide.
- It embeds guidance on many modern techniques and approaches: object technology and component-based development, modeling and UML, architecture, iterative development, and so on.
- It is an open process framework that software development organizations can configure and extend to suit their own needs.
- It is not a frozen product; rather, it is alive, constantly maintained, and continuously evolving, with new process components made available online.
- It is based on a solid process architecture, and that allows a development organization to configure and tailor it to suit its needs.
- It supports key best practices for software development.
- It is supported by an extensive palette of tools developed by IBM Rational.