PART I

INTRODUCING THE RATIONAL UNIFIED PROCESS

CHAPTER 1

Introducing the Rational Unified Process

What Is the Rational Unified Process?

When you ask this question, you typically get different answers, depending on whom you ask and the context of your question. What makes matters confusing is that the Rational Unified Process, or the RUP, actually denotes three very different things:

- The RUP is a **software development approach** that is iterative, architecture-centric, and use-case-driven. It is described in a variety of whitepapers and books. The most comprehensive information can be found in the RUP product itself, which contains detailed guidelines, examples, and templates covering the full software lifecycle.
- The RUP is a **well-defined and well-structured software engineering process**. It clearly defines who is responsible for what, how things are done, and when to do them. The RUP also provides a well-defined structure for the lifecycle of a RUP project, clearly articulating essential milestones and decision points.
- The RUP is also a **process product** that provides you with a **customizable process framework** for software engineering. The RUP product supports process customization and authoring, and a wide variety of processes, or Process Configurations, can be assembled

from it. These RUP configurations can be made to support small or large teams and disciplined or less-formal approaches to development. The RUP product contains several out-of-the-box Process Configurations and Process Views that guide analysts, developers, testers, project managers, configuration managers, data analysts, and other team members in how to develop software. The RUP is used by a wide variety of companies in different industry sectors.

In this chapter we will get a better understanding of what the RUP is by elaborating on each of these three perspectives of the RUP.

The RUP—The Approach

In this section we discuss the essential principles the RUP uses to facilitate successful software development, as well as the keystone iterative approach to applying those principles.

Underlying Principles of the RUP Approach

At the core of the Rational Unified Process lie several fundamental principles that support successful iterative development and that represent the essential "Spirit of the RUP" (which is discussed in Chapter 2). These principles were gleaned from a huge number of successful projects and distilled into a few simple guidelines. These principles are listed here:

- **Attack major risks early and continuously. . . or they will attack you.**[1] Rather than addressing business risks, technical risks, or other risks later in a project, identify and attack major risks as early as possible.
- **Ensure that you deliver value to your customer.** Document the requirements in a form that is easily understood by customers, but work closely to the requirements through design, implementation, and testing phases to ensure that you also deliver the requirements.

1. See Gilb 1988.

The Spirit of the RUP

Essential Principles

- Attack major risks early and continuously…or they will attack you.
- Ensure that you deliver value to your customer.
- Stay focused on executable software.
- Accommodate change early in the project.
- Baseline an executable architecture early on.
- Build your system with components.
- Work together as one team.
- Make quality a way of life, not an afterthought.

- **Stay focused on executable software.** Documents, designs, and plans are all good, but they are a poor indication of true progress because their evaluation is subjective and their importance is secondary compared to the code itself. Executable code that compiles and successfully passes tests is the best indication of progress.
- **Accommodate change early in the project.** Today's applications are too complex to enable us to get the requirements, design, and implementation correct the first time. This means that to develop a good enough system, we need to allow and adapt to change.
- **Baseline an executable architecture early on.** Many project risks can be mitigated by designing, implementing, and testing the architecture early in the project. Establishing a stable architecture early on also facilitates communication and localizes the impact of changes.
- **Build your system with components.** Applications built with components are more resilient to change and can have radically reduced system maintenance cost. Components facilitate reuse, allowing you to build higher quality applications faster than using functional decomposition.

- **Work together as one team.** Software development has become a team sport,[2] and an iterative approach emphasizes the importance of good team communications and a team spirit where each team member feels responsible for the completed end product.
- **Make quality a way of life, not an afterthought.** Ensuring high quality involves more than just the testing team. It involves *all* team members and *all* parts of the lifecycle. An iterative approach focuses on early testing and test automation for more effective regression testing, thus reducing the number of defects.

These principles are covered in more detail in Chapter 2.

The RUP and Iterative Development

Most software teams still use a **waterfall** process for development projects, where they complete each phase in a strict sequence of requirements, then analysis and design, then implementation/integration, and then testing. Or, more commonly, a modified waterfall approach with feedback loops added to the basic overall flow just described. Such approaches leave key team members idle for extended periods of time and defer testing until the end of the project lifecycle, when problems tend to be tough and expensive to resolve, and pose serious threats to release deadlines.

By contrast, the RUP uses an **iterative** approach (iterate = repeat), that is, a sequence of incremental steps or iterations. Each iteration includes some, or most, of the development disciplines (requirements, analysis, design, implementation, and so on), as you can see in Figure 1.1. Each iteration has a well-defined set of objectives and produces a partial working implementation of the final system. Each successive iteration builds on the work of previous iterations to evolve and refine the system until the final product is complete.

Early iterations will have a greater emphasis on requirements and analysis and design; later iterations will have a greater emphasis on implementation and testing.

The iterative approach has proven itself superior to the waterfall approach for a number of reasons:

---

2. See Booch 2001.

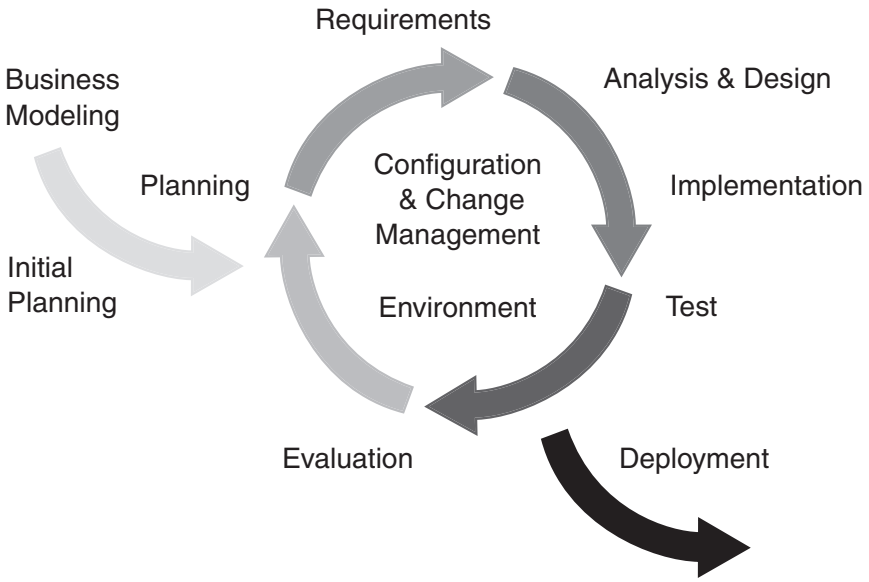

FIGURE 1.1 Iterative Development in the RUP. *The Rational Unified Process promotes an iterative approach—in each iteration you do a little requirements, analysis, design, implementation, and testing. Each iteration builds on the work of the previous iterations to produce an executable that is one step closer to the final product.*

- **It accommodates changing requirements.** Requirements change and "feature creep"—the addition of features that are technology- or customer-driven—have always been primary sources of project trouble, leading to late delivery, missed schedules, dissatisfied customers, and frustrated developers. Instead, iterative development focuses the team on producing and demonstrating executable software in the next few weeks, which forces a focus on the most essential requirements.
- **Integration is not one "big bang" at the end of a project.** Leaving integration to the end results in time-consuming rework—sometimes up to 40 percent of the total project effort. To avoid this, an iterative approach breaks a project down into smaller iterations, each ending with an integration in which building blocks are integrated progressively and continuously, minimizing later rework.

- **Risks are usually discovered or addressed during early integrations.** The RUP's integrated approach mitigates risks in early iterations, where all process components are tested. Since each iteration exercises many aspects of the project, such as tools, off-the-shelf software, and team members' skills, you can quickly discover whether perceived risks prove to be real and also uncover new, unsuspected risks at a time when they are easier and less costly to address.
- **Management has a means of making tactical changes to the product—**to compete with existing products, for example. Iterative development quickly produces an executable architecture (albeit of limited functionality), which can be used for quick release of a product with a reduced scope to counter a competitor's move.
- **Reuse is facilitated.** It is easier to identify common parts as they are being partially designed or implemented in iterations, than to recognize them during planning. Design reviews in early iterations allow architects to spot potential opportunities for reuse and then develop and mature common code for these opportunities in subsequent iterations.
- **Defects can be found and corrected over several iterations,** resulting in a robust architecture and a high-quality application. Flaws are detected even in early iterations rather than during a 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.
- **It is a better use of project personnel.** Many organizations match their use of a waterfall approach with a pipeline organization: The analysts send the completed requirements to designers, who send a completed design to programmers, who send components to integrators, who send a system to testers. These many handoffs are sources of errors and misunderstandings and make people feel less responsible for the final product. An iterative process widens the scope of expertise of the team members, allowing them to play many roles and enabling a project manager to use the available staff better, while simultaneously removing harmful handoffs.
- **Team members learn along the way.** The project members have several opportunities along a development cycle to learn from

their mistakes and to improve their skills from one iteration to another. More training opportunities can be discovered as the result of assessing the earlier iterations. In contrast, in a waterfall approach, you have only one shot at design or coding or testing.

- **The development process itself is 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 or scheduling perspective, but also analyzes what, in both the organization and the process, can be improved in the next iteration.

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

The RUP—A Well-Defined Software Engineering Process

Modeling of the RUP is based on the Software Process Engineering Metamodel (SPEM).

The Rational Unified Process itself was designed with techniques similar to those used in software design. In particular, it is modeled using the Software Process Engineering Metamodel (SPEM)[3]—a standard for process modeling based on the Unified Modeling Language (UML).[4] Figure 1.2 shows the overall architecture of the Rational Unified Process. The process has two structures or, if you prefer, two dimensions:

- **Dynamic structure.** The horizontal dimension represents the **dynamic structure** or time dimension of the process. It shows how the process, expressed in terms of cycles, phases, iterations, and milestones, unfolds over the lifecycle of a project

3. See OMG 2001.

4. A standard for specifying, visualizing, and documenting models of software systems, including their structure and design. The standard is managed by the Object Management Group (www.omg.org).

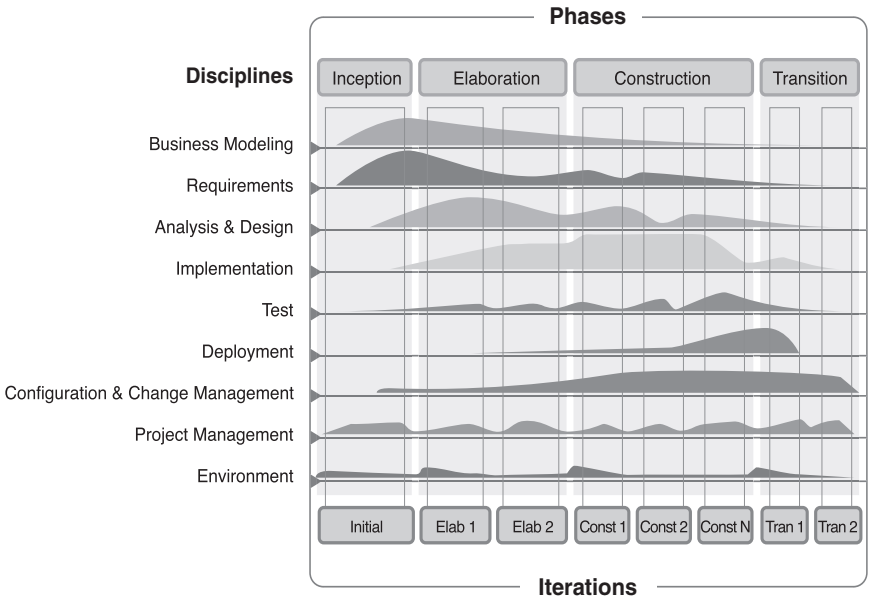

FIGURE 1.2 Two Dimensions of the RUP. *The Rational Unified Process is organized along two dimensions: The dynamic aspect (horizontal) expresses cycles, phases, iterations, and milestones; the static aspect (vertical) expresses activities, disciplines, artifacts, and roles.*

- **Static structure.** The vertical dimension represents the **static structure** of the process. It describes how process elements—activities, disciplines, artifacts, and roles—are logically grouped into core process disciplines (or workflows).

These dimensions are discussed in the following sections.

The Dynamic Structure of the Rational Unified Process

The dynamic structure deals with the lifecycle or time dimension of a project. The RUP provides a structured approach to iterative development, dividing a project into four **phases:** Inception, Elaboration, Construction, and Transition (see Figure 1.3).[5] Chapters 6 through 9 discuss

5. See Kruchten 2000a.

these phases in detail. The objectives and milestones of the phases are listed in the sidebar RUP Lifecycle Phases, Objectives, and Milestones.

RUP Lifecycle Phases, Objectives, and Milestones

Inception Phase

Objectives:

- Understand the scope of the project
- Build the business case
- Get stakeholder buy-in to move ahead

Milestone: Lifecycle Objective Milestone (LCO)

Elaboration Phase

Objectives:

- Mitigate major technical risks
- Create a baselined architecture
- Understand what it takes to build the system

Milestone: Lifecycle Architecture Milestone (LCA)

Construction Phase

Objective:

- Build the first operational version of the product

Milestone: Initial Operational Capability Milestone (IOC)

Transition Phase

Objective:

- Build the final version of the product and deliver it to the customer

Milestone: Product Release Milestone (PR)

Each phase contains one or more **iterations**, which focus on producing the technical deliverables necessary to achieve the business objectives of that phase. There are as many iterations as it takes to address the objectives of that phase sufficiently, but *no more*. If objectives can't be addressed within the planned phase, another iteration should be added to the phase—which will delay the project. To avoid this, make

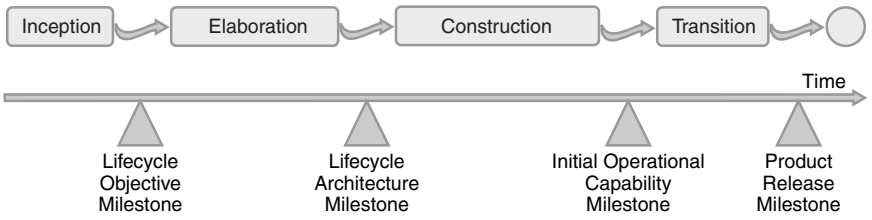

FIGURE 1.3 Milestones for the RUP Lifecycle Phases. *Each of the RUP's four phases has a milestone and a well-defined set of objectives. Use these objectives as a guide for deciding which activities to carry out and which artifacts to produce.*

Each iteration is sharply focused on just *what is needed to achieve the business objectives of the phase.*

sure that each iteration is sharply focused on *just* what is needed to achieve the business objectives of that phase. For example, focusing too heavily on requirements in Inception is counterproductive. Chapter 12 discusses project planning.

- **Inception.** Establish a good understanding of what system to build by getting a high-level understanding of all the requirements and establishing the scope of the system. Mitigate many of the business risks, produce the business case for building the system, and get buy-in from all stakeholders on whether to proceed with the project.
- **Elaboration.** Take care of many of the most technically difficult tasks: Design, implement, test, and baseline an executable architecture, including subsystems, their interfaces, key components, and architectural mechanisms, such as how to deal with inter-process communication or persistency. Address major technical risks, such as resource contention risks, performance risks, and data security risks, by implementing and validating actual code.
- **Construction.** Do most of the implementation as you move from an executable architecture to the first operational version of your system. Deploy several internal and alpha releases to ensure that the system is usable and addresses user needs. End the phase by deploying a fully functional beta version of the system, including

installation and supporting documentation and training material (although the system will likely still require tuning of functionality, performance, and overall quality).

- **Transition.** Ensure that software addresses the needs of its users. This includes testing the product in preparation for release and making minor adjustments based on user feedback. At this point in the lifecycle, user feedback focuses mainly on fine-tuning the product, configuration, installation, and usability issues; all the major structural issues should have been worked out much earlier in the project lifecycle.

The Static Structure of the Rational Unified Process

The static structure deals with how process elements—activities, disciplines, artifacts, and roles—are logically grouped into core process disciplines. A process describes **who** is doing **what, how,** and **when.** As shown in the sidebar Four Key Modeling Elements of the RUP and Figure 1.4, the Rational Unified Process is represented using four key modeling elements.

Four Key Modeling Elements of the RUP

- **Roles.** The *who*
- **Activities.** The *how*
- **Artifacts.** The *what*
- **Workflows.** The *when*

Roles

A **role** is like a "hat" that an individual (or group) wears during a project. One individual may wear many different hats. This is an important point because it is natural to think of a role as an individual on the team, or as a fixed job title, but in the RUP the roles simply define how the individuals should do the work, and they specify the

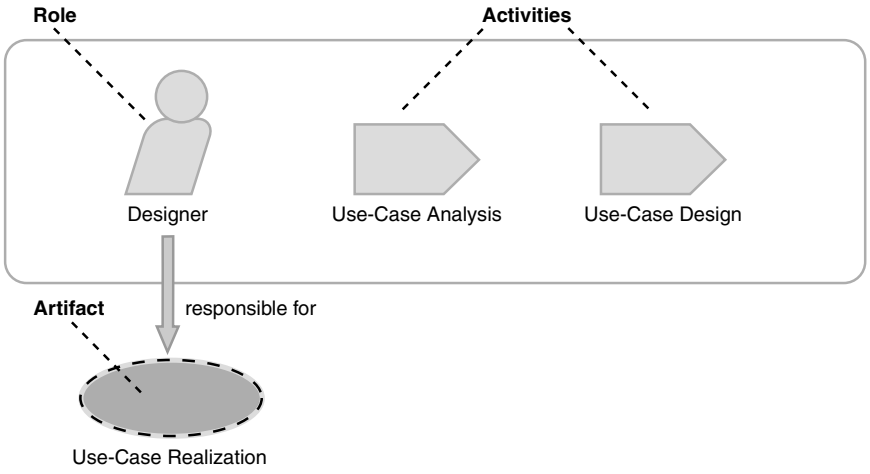

FIGURE 1.4 Roles, Activities, and Artifacts. *A* role *expresses who (an individual or a group) is doing the work; an* activity *describes how the work is done; and an* artifact *captures what is done.*

competence and responsibility that the individual(s) playing that role should have. A person usually performs one or more roles, and several people can perform the same role.

Activities

An **activity** of a specific role is a unit of work that an individual in that role may be asked to perform. The activity has a clear purpose, usually expressed in terms of creating or updating some artifacts, such as a model, a component, or a plan. Each activity is assigned to a specific role. An activity generally takes a few hours to a few days to complete, usually involves one person, and affects one or only a small number of artifacts. An activity should be usable as an element of planning and progress; if it is too small, it will be neglected, and if it is too large, progress would have to be expressed in terms of an activity's parts. Activities may be repeated several times on the same arti-

fact—especially when going from one iteration to another, refining and expanding the system—by the same role, but not necessarily by the same individual.

Steps

Activities are broken down into **steps,** which fall into three main categories:

- **Thinking** steps, where the person playing the role understands the nature of the task, gathers and examines the input artifacts, and formulates an outcome
- **Performing** steps, where the role creates or updates some artifacts
- **Reviewing** steps, where the role inspects the results against some criteria

Not all steps are necessarily performed each time an activity is invoked, so steps can be expressed in the form of alternate flows.

Artifacts

An **artifact** is a piece of information that is produced, modified, or used by a process. Artifacts are the tangible project elements: things the project produces or uses while working toward the final product. Artifacts are used as input by roles to perform an activity and are the result or output of other activities.

Artifacts may take various shapes or forms:

- A **model,** such as the Use-Case Model or the Design Model
- A **model element,** that is, an element within a model, such as a class, a use case (UC), or a subsystem
- A **document,** such as the Vision or Business Case
- Source code
- **Executables,** such as an executable **Prototype**

An artifact can be documented formally (using a tool) or informally (captured in an e-mail message or on a whiteboard). This will be discussed in more detail in Chapter 3 and Chapters 6–9.

Workflows

A mere listing of all roles, activities, and artifacts does not quite constitute a process. You need a way to describe meaningful sequences of activities that produce some valuable result and to show interactions between roles—this is exactly what **workflows** do.

Workflows come in different shapes and forms; the two most common workflows are **Disciplines,** which are high-level workflows (see the section Disciplines that follows), and **Workflow Details,** which are workflows within a discipline.

People are not machines, and the workflow cannot be interpreted literally as a program for people to be followed exactly and mechanically.

In UML terms, a workflow can be expressed as a sequence diagram, a collaboration diagram, or an activity diagram. Figure 1.5 shows an example workflow. Note that it is not always possible or practical to represent all of the dependencies between activities. Often two activities are more tightly interwoven than shown, especially when carried out by the same individual. People are not machines, and the workflow cannot be interpreted literally as a program for people to be followed exactly and mechanically.

Additional Process Elements

Roles, activities (organized in workflows), and artifacts represent the backbone of the Rational Unified Process static structure. But there are some other elements added to activities or artifacts that make the process easier to understand and use and that provide more complete guidance to the practitioner. These additional process elements are

- **Guidelines,** to provide rules, recommendations, or heuristics that support activities, steps, and artifacts.
- **Templates,** for the various artifacts.
- **Tool mentors,** to establish a link with and provide guidance on using the development tools.
- **Concepts,** to introduce key definitions and principles.
- **Roadmaps,** to guide the user into the RUP from a given perspective.

Figure 1.6 shows how these elements enhance the primary elements.

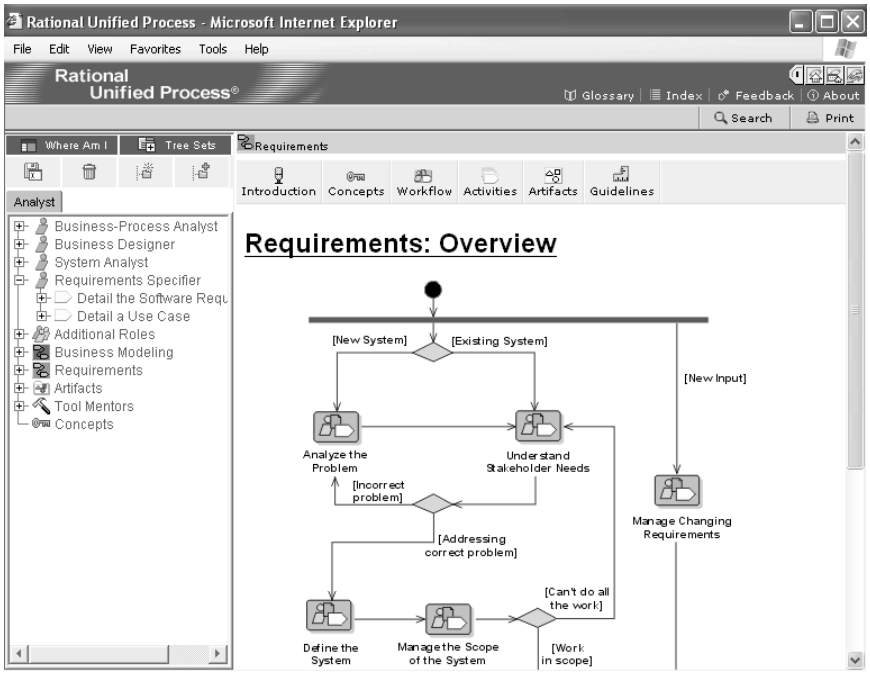

FIGURE 1.5 The Workflow of the Requirements Discipline. *A workflow shows in what order to carry out activities to accomplish something of measurable value. RUP workflows typically show only the most essential dependencies between activities to avoid cluttering the view.*

Disciplines

Finally, all process elements—roles, activities, artifacts, and the associated concepts, guidelines, and templates—are grouped into logical containers called **Disciplines**. There are nine disciplines in the standard RUP product (see the RUP Disciplines sidebar).

This list is not definitive, and any company doing extensions to the RUP product could introduce additional disciplines.

RUP Disciplines

- Business modeling
- Requirements management
- Analysis and design
- Implementation
- Deployment
- Test
- Project management
- Change management
- Environment

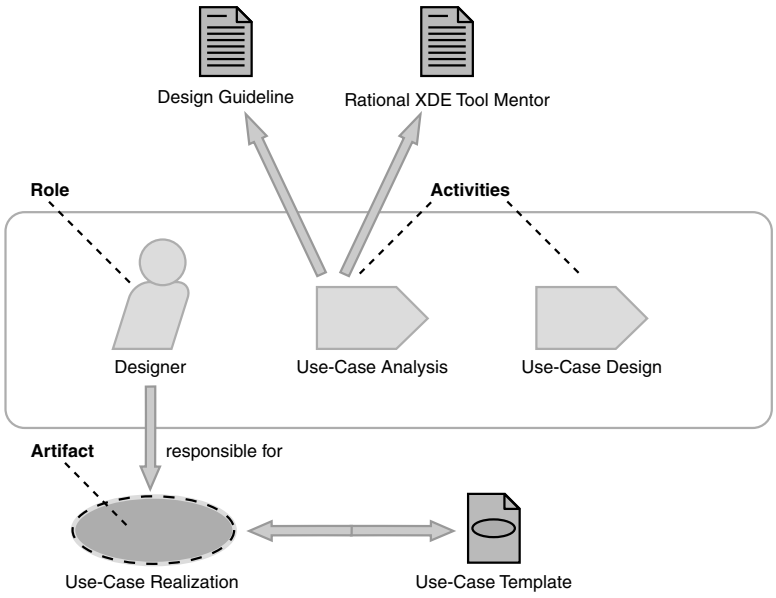

FIGURE 1.6 Adding Templates, Tool Mentors, and Guidelines. *Templates jump-start the production of an artifact; tool mentors provide detailed guidance for carrying out an activity, or step, using the tools at hand; and guidelines provide detailed guidance on activities, steps, or artifacts.*

The RUP—A Customizable Process Product

The RUP product constitutes a complete process framework.

Each project and each organization have unique needs requiring a process that is adapted to their specific situation. To accommodate this requirement, the RUP product constitutes a complete process framework (see Figure 1.7) composed of several integrated parts:

- **Best practices.** The RUP comes with a library of best practices, produced by IBM Software and its partners. These best practices are continually evolving and cover a broader scope than this book can cover. The best practices are expressed in the form of phases, roles, activities, artifacts, and workflows.

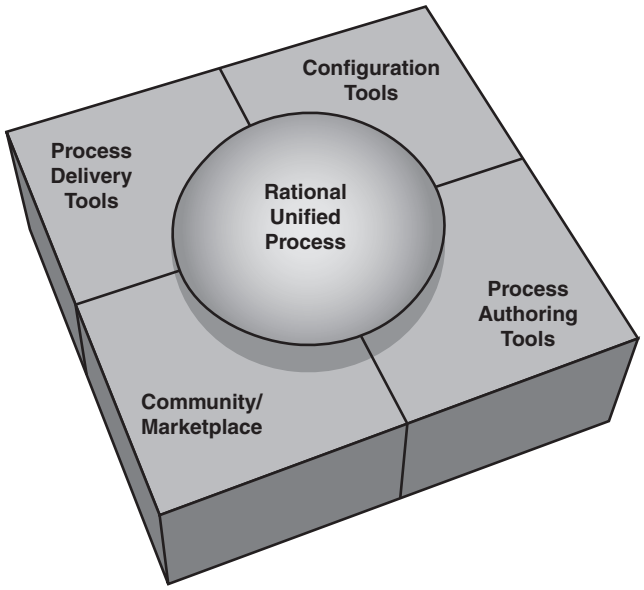

FIGURE 1.7 The RUP Process Framework. *The RUP product consists of a broad set of best practices, configuration tools for selecting appropriate best practices, process delivery tools for accessing these best practices, an online community for exchanging artifacts and experiences, and process authoring tools for adding your own best practices to the RUP.*

- **Process delivery tools.** The RUP is literally at the developers' fingertips because it is delivered online using Web technology, rather than using books or binders. This delivery allows the process to be integrated with the many software development tools in the Rational Suite and with any other tools so developers can access process guidance within the tool they are using.
- **Configuration tools.** The RUP's modular and electronic form allows it to be tailored and configured to suit the specific needs of a development organization. This is discussed briefly in the section Configuration and Process Authoring Tools, and in more detail in Chapter 10.
- **Process authoring tools.** Rational Process Workbench (RPW) is a process authoring tool, allowing you to capture your own best practices into the RUP format (see Chapter 10 for details).
- **Community/Marketplace:** The Rational Developer Network (RDN) online community allows users to exchange experiences with peers and experts, and provides access to the latest information in terms of artifacts, articles, or additional RUP content.

The RUP product is designed, developed, delivered, and maintained like any software tool. It is upgraded approximately twice a year, so the process is never obsolete and its users benefit from the latest developments.

Let's take a closer look at two areas of the RUP's capabilities: configuration and process authoring tools and process delivery tools.

Configuration and Process Authoring Tools

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 still fulfilling its mission to help developers rapidly produce predictably high-quality software. An adopting organization's best practices, along with its specific rules and procedures, should complement the process.

Since the Rational Unified Process is a **process framework,**[6] it can be adapted and extended to suit the needs of an adopting organization. The RUP framework is made up of components, divided into a base unit and a series of RUP Plug-Ins. As shown in Figure 1.8, you can produce a Process Configuration by selecting a set of RUP Plug-Ins to add to the base. You can even make a fine granular selection of which process components to deploy within your plug-ins and base. This makes it easy to create a Process Configuration that addresses the spe-

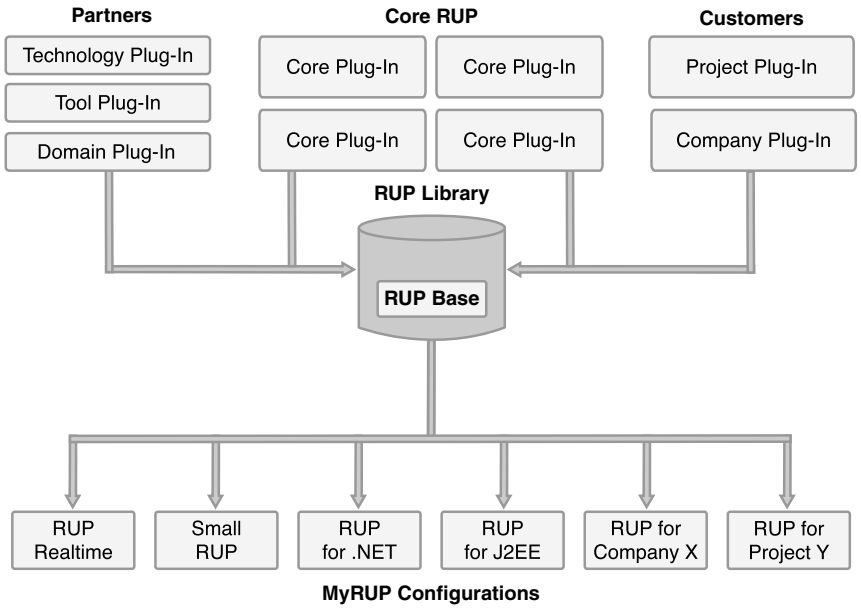

FIGURE 1.8 The RUP's Component-Based Architecture Allows the RUP to Be Configured to Project Needs. *The component-based architecture of the RUP product allows RUP users to choose from a wide variety of plug-ins and to deploy those that are appropriate for their project.*

6. See Kroll 2001.

cific needs, characteristics, constraints, culture, and domain of your project or organization.

You can also produce different views of the process, so-called Process Views, to allow each person in your project to view the Process Configuration from a perspective that is relevant to his or her role and responsibilities. The RUP product also comes with a number of out-of-the-box configurations and process views that you can use as a starting point. Chapter 10 details how to configure the RUP product.

Companies using the RUP process authoring tool can package their know-how of a certain technology, tool, or domain into a RUP Plug-In and make it available to other RUP users inside or outside their organization. RUP users can also produce plug-ins that are specific for a project, a division, or their entire organization, and thus make their know-how easily accessible to all their software engineers. When configuring the RUP, projects can take advantage of the best practices captured in these partner and in-house plug-ins and create configurations that fit their specific needs. Chapter 10 details how to build RUP Plug-Ins.

Process Delivery Tools

Using a process can be complicated; the RUP product helps users by providing MyRUP (a Web interface that can be personalized), tool mentors, and Extended Help. The combination of tool mentors and Extended Help provides a two-way integration between the RUP and the tools at your desktop. This integration helps practitioners make more effective use of their tools, allowing them to get more value out of their tool investment and facilitating effective implementation of the process.

MyRUP provides a role-based tree browser containing links to the parts of your project's RUP Process Configuration that are relevant for you.

MyRUP

The RUP product can be delivered through a personalized Web browser, called MyRUP. MyRUP provides Process Views, a role-based or personalized tree control containing links to the parts of your project's RUP Process Configuration that are relevant for you, as well as links to files or URLs external to your configuration (see Figure 1.9).

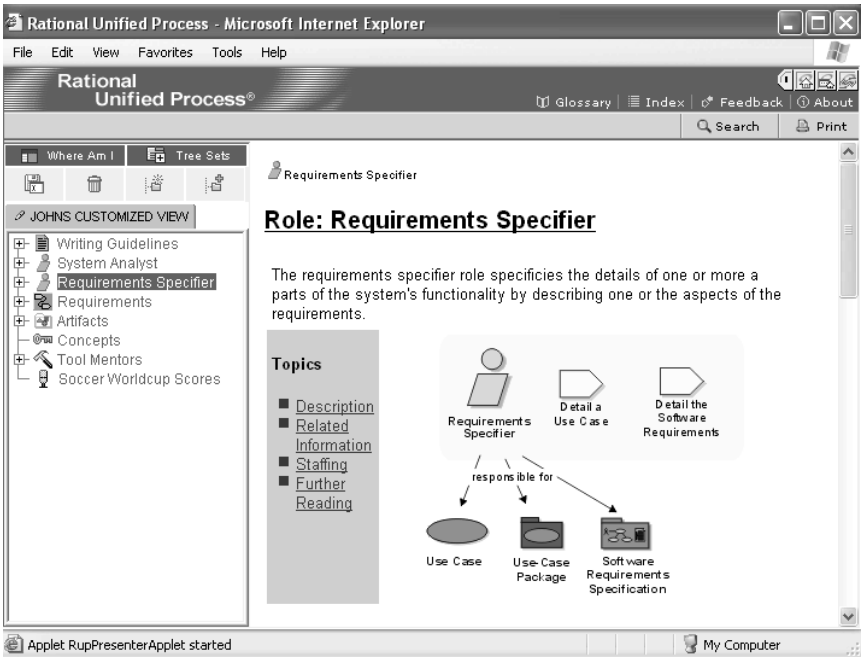

FIGURE 1.9 MyRUP Provides Personalized Views. *MyRUP is a personalized Web interface with the RUP that allows users to find information easily through a personalized view, search engine, graphical navigation, and tree control.*

Each user can customize his or her own Process View or use one of the predefined Process Views. MyRUP also provides a search engine, a glossary, and a Getting Started tutorial.

Tool Mentors

The bulk of the RUP product is tool-independent, although many of the RUP activities need to be carried out using various tools, and practitioners need to understand how to implement the process with the tools at hand. **Tool mentors** provide step-by-step guidelines for implementing the various RUP activities using the tools at hand. The

tool mentors describe which menus to select, what to enter in dialog boxes, and how to draw diagrams to accomplish the specified tasks.

Tool mentors are available for Rational tools, as well as for other tools such as IBM WebSphere Application Server and BEA WebLogic. Customers and partners can write additional tool mentors, and tool mentors can be included in RUP Plug-Ins, as can any process element in the RUP product.

Extended Help

Extended Help provides context-sensitive process guidance within the various tools. For example, if you are trying to use the Class Diagram editor in Rational Rose and you do not know what to do next, you can open Extended Help from the Rose tools menu. This will give you a list of the most relevant topics within the RUP, depending on the context, in this case, a Class Diagram in Rose, see Figure 1.10.

Who Uses the RUP Product?

Roughly 10,000 companies are using the RUP product. They use it in various application domains, evenly distributed over both large and small projects. This variety shows the versatility and wide applicability of the RUP product. Here are examples of the various industry sectors around the world that use it:

- Telecommunications
- Transportation, aerospace, defense
- Manufacturing
- Financial services
- Systems integrators

The RUP has become widely adopted over the last few years, which is a sign of change in our industry. As time-to-market pressure increases, as well as the demand for high-quality applications, companies are looking to learn from others' experience and are ready to adopt proven best practices.

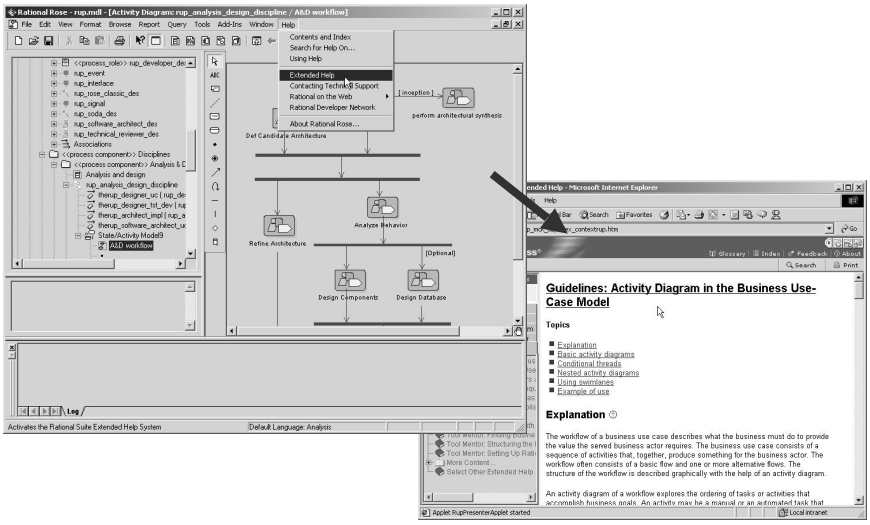

FIGURE 1.10 RUP Context-Sensitive Extended Help. *Extended Help provides context-sensitive help from the tools being used. When launched, it presents a list of the most relevant topics in the RUP product.*

The way these organizations use the Rational Unified Process varies greatly. Some use it very formally and with a high degree of discipline; they have evolved their own company process from the RUP product, which they follow with great care. Other organizations have a more informal usage, taking the RUP platform as a repository of advice, templates, and guidance that they use as they go along, as a sort of knowledge base on software engineering.

By working with these customers, observing how they use the RUP product, listening to their feedback, and looking at the additions they make to the process to address specific concerns, the RUP development team at IBM Software continues to refine the RUP product for the benefit of all.

Conclusion

We discussed that the Rational Unified Process, or the RUP, denotes three very different things:

- The RUP is an **approach for developing software,** which is iterative, architecture-centric, and use-case-driven. It is described in whitepapers and books, and the most complete set of information about the RUP approach can be found in the RUP product itself.
- The RUP is a **well-defined and well-structured software engineering process.** It clearly defines project milestones, who is responsible for what, how things are done, and when you should do them.
- The RUP is also a **process product** that provides you with a **customizable process framework** for software engineering. You can configure the RUP product to support small or large teams and disciplined or less-formal approaches to development. It also allows you to add your own best practices and to share experiences and artifacts with peers and experts.

Now that we have a better idea of what the RUP is, let us go deeper to understand its essence and its spirit, that is, the fundamental principles behind the RUP.