CHAPTER 1

Welcome to the IBM Rational Unified Process and Certification

By Ahmad K. Shuja

This chapter is about RUP's breadth and not its depth, so this introductory chapter will be mile wide and inch deep in terms of content. It introduces some of the core concepts that the RUP is founded on but does not go into detail about its different parts. It does, however, discuss the most important aspects of the RUP as they relate to software development. This chapter provides an overview of the book content.

An Overview of the Rational Unified Process

The IBM Rational Unified Process, also known as the RUP, is a process framework for successful iterative-incremental software development. In the software engineering domain, there are a number of development methodologies that organizations have successfully adapted and adopted to meet specific business needs. These range from traditional waterfall development to more agile ones. Figure 1-1 shows some of the more famous methodologies and where each can be positioned with respect to agility and discipline. Note that the up-front goals modeling component may not be directly associated with any given software development methodology but is there to ensure alignment between new software products or releases and the business strategy.

At its core, RUP is defined by the following three central elements:

- Key principles for business-driven development
- A framework of reusable method content and process building blocks
- The underlying method and process definition language

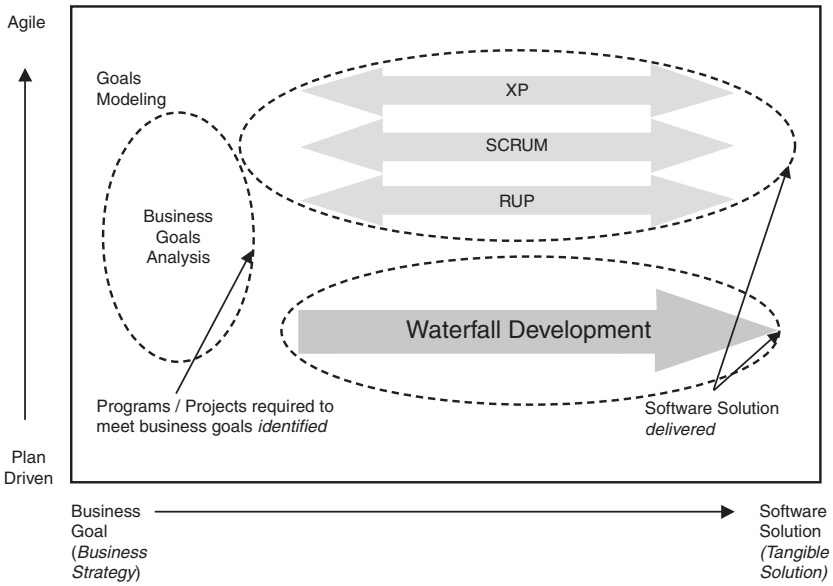


Figure 1-1 Methodologies map

Key Principles

RUP focuses on six key principles in software engineering (formerly known as *best practices*). These principles, which are easy to memorize because they start with the letters *A* through *F*, constitute the foundation of the RUP:

- **A**dapt the process.
- **B**alance stakeholder priorities.
- **C**ollaborate across teams.
- **D**emonstrate value iteratively.
- **E**levate the level of abstraction.
- **F**ocus continuously on quality.

The key principals are not sequential; in fact, you will see in Chapter 2, "Key Principles for Business-Driven Development," that the principles actually reinforce each other. For example, the principle of demonstrating value iteratively supports the principle of focusing continuously on quality. Similarly, other key principles support and drive one another. Chapter 2 explains all six key principles in detail and discusses their inter-relationships.

A Framework of Reusable Method Content and Process Building Blocks

A **process framework** can be defined as an incomplete support structure in which another process can be organized and developed. Therefore, you need to finish a process framework

before you can apply it to specific projects within an organization. Similarly, you need to finish the RUP skeleton and its libraries to fit the organization.

The RUP framework is defined by a family of method plug-ins from which, based on the unique business needs as well as the context (technical and management complexity), organizations are able to create their own method configurations and tailored processes. RUP provides an architectural foundation and wealth of material from which a process definition can be constructed, therefore enabling the adopting organization to configure and extend that foundation as desired.

A few factors influence the configuring and tailoring of RUP:

- Project complexity

  In most cases, the more complex and technical the project, the greater the formality and control required to ensure its successful completion and timely delivery. This formality normally involves greater plan-driven development and more discipline. The term commonly used in RUP to determine the level of formality and control that is required within the process is **ceremony**. Figure 1-2 shows the relationship between complexity and ceremony. Accordingly, the level of ceremony affects the number of artifacts and details of the workflow descriptions.

- Organizational maturity

  Less mature organizations might require more discipline than more mature ones.

- Organization culture

  Culture plays an important role in the successful adaptation and adoption of the process.

- Regulatory compliance and policy requirements

  Some industries, especially financial and healthcare, might require more controls, which in turn require a high ceremony process and more artifacts.

- Development type

  The type of software development, such as green field versus COTS based, affects the process.

- Organization size

  The size of the organization determines how to customize the RUP to enable successful development and timely delivery of the software solutions.

These factors lead to one or more RUP flavors meeting the specific needs of an organization. IT organizations commonly develop multiple RUP instances to meet the needs of different types of projects. That approach satisfies the need for different levels of ceremony for small or large IT projects. In Part IV, we discuss Rational Method Composer, which can be used for effectively and efficiently customizing and publishing various flavors of RUP. We include some further discussion within Chapter 13, “Environment.” Even though hundreds, if not thousands,

of RUP customizations have been performed, they are based on the original RUP process framework, which is the subject of this book and the certification.

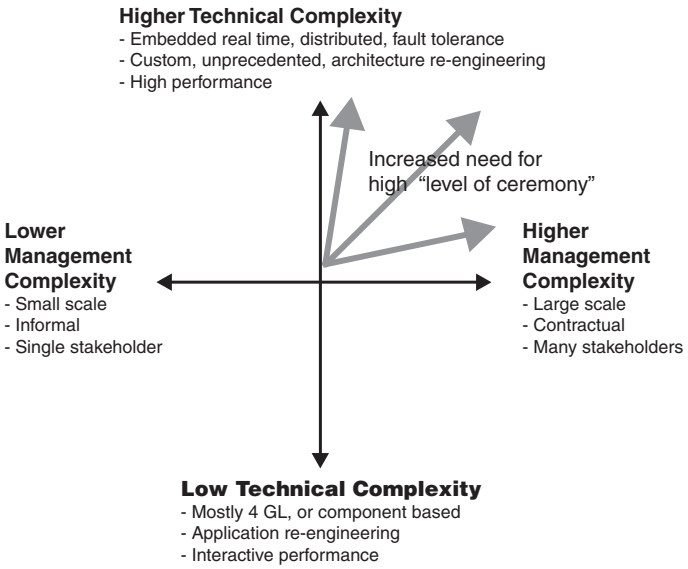


Figure 1-2 Complexity and ceremony (source: IBM Rational Unified Process v7.0)

Architectural Views

RUP represents the software architecture in multiple architectural views. Each architectural view addresses concerns specific to stakeholders in the development process. These stakeholders might include users, designers, managers, and maintainers. The architectural views capture the major design decisions by presenting the software architecture in terms of how components connect to produce useful forms (Perry & Wolf, 1992).

The typical set of views in the RUP, called the **4+1 view model**, is composed of the following.

- Use-Case view

  This view provides a basis for planning the technical content of iterations. It is used in the Requirements discipline.

- Logical view

  This view provides a basis for understanding the structure and organization of the design of the system. Logical view is used in the Analysis and Design discipline.

- Implementation view

  This view captures the enumeration of all subsystems in the Implementation Model, the component diagrams illustrating how subsystems are organized in layers, and hierarchies and illustrations showing important dependencies between subsystems.

- Process view

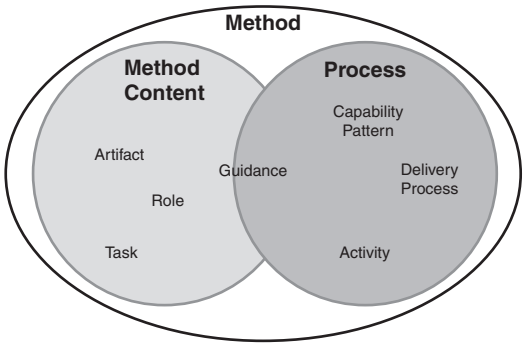
  This view illustrates the process decomposition of the system, including the mapping of classes and subsystems on to processes and threads. The Process view is used in the Analysis and Design discipline.

- Deployment view

  This view illustrates the distribution of processing across a set of nodes in the system, including physical distribution of processes and threads. This view is used in the Analysis and Design discipline.

Method and Process Definition Language

A unified method architecture (UMA) meta-model provides a language for describing method content and processes. UMA is an architecture to conceive, specify, and store method and process metadata. UMA clearly separates Method Content definitions from their application in delivery processes. It does this by defining the reusable core Method Content in the form of general content descriptions and the project-specific applications in the form of process descriptions. Basic elements of UMA are shown in Figure 1-3. We will discuss UMA is greater detail in Part II, "Unified Method Architecture (UMA)."



IBM Rational Unified Process. The diagram was used publicly in a *Rational Edge* article in 2006.

Figure 1-3 The basic elements of UMA

An Overview of the RUP Architecture

This valuable section offers a concise explanation of the RUP architecture. It starts with the popular hump chart, shown in Figure 1-4, which illustrates the overall RUP architecture. This figure contains information on phases, iterations, milestones, disciplines, their inter-relationships, and the lifecycle concept. This section focuses primarily on establishing a foundation from which you will be able to achieve the most value from this book. This foundation will enable you to clearly appreciate the relationships between different components of RUP architecture, as illustrated in Figure 1-4. By the end of this section, you will look at the RUP hump chart differently; you will be able to discuss why some disciplines should be there in your organization's specific rollout of

RUP and how phases, iterations, milestones, and disciplines are related. This chapter also discusses the importance of disciplines, phases, and iterations in iterative and incremental development. Let's now discuss these components in a little more detail.

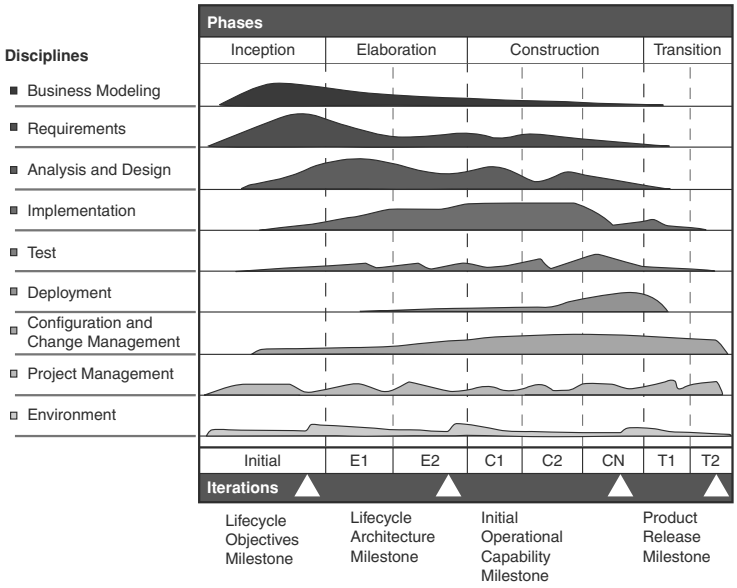


Figure 1-4 The Rational Unified Process overview

Phases and Milestones

The RUP provides an iterative and incremental approach to developing software. This iterative and incremental development happens within iterations that occur within a structured lifecycle consisting of phases and milestones. The RUP has four sequential phases: Inception, Elaboration, Construction, and Transition. Each of them plays a central role in managing iterative[1] and incremental development projects using RUP. Each phase concludes with a major milestone, as shown in Figure 1-5. The following sections provide a brief discussion of each phase. The focus in this section is to clarify, at the process framework level, the difference between the traditional waterfall lifecycle and the iterative and incremental development lifecycle as implemented by the RUP.

Overview

As discussed earlier, phases are made up of iterations, and both phases and iterations are important concepts to grasp for building a concrete understanding of the RUP. Disciplines play an important role in designing the iterations carried out within each phase. Although we do discuss disciplines later, let's briefly see how the RUP defines the term *discipline*. According to the RUP, "a discipline is

[1] Iterative development is an approach to building a product (software or any other product) in iterations. Each iteration is a small project with its own clear deliverables. In software projects, most iterations end up with an executable build.

a collection of related activities that are related to a major area of concern." Based on which phase you are in, each iteration contains activities from across different disciplines. For instance, earlier in the RUP lifecycle, although there is a focus on the development of an executable build as early in the lifecycle as possible, there is a greater need to build an understanding of the business problem or opportunity. Therefore, this need, as shown in the hump chart, requires more activities from the Business Modeling and Requirements disciplines to be performed earlier in the RUP project lifecycle.

Iterations are designed and executed with certain goals in mind. Depending upon which phase the iteration belongs to, the iteration goals are aligned to accomplish the respective milestone. For example, iterations for the Elaboration phase are designed such that its objectives are achieved and the Lifecycle Architecture milestone is accomplished. Therefore, achievement of each iteration goal moves the project closer to achieving the respective objectives of that phase. This concept is presented in Figure 1-6. Each iteration has its own respective goals and is designed such that collectively, the iterations are executed within a given phase. These iterations achieve respective objectives of a milestone.

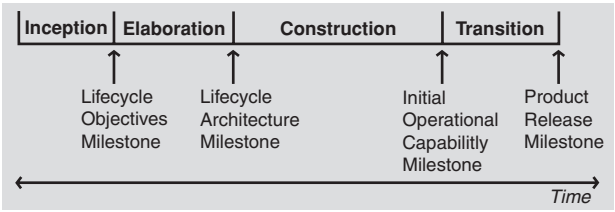


Figure 1-5 The phases and milestones of a RUP project

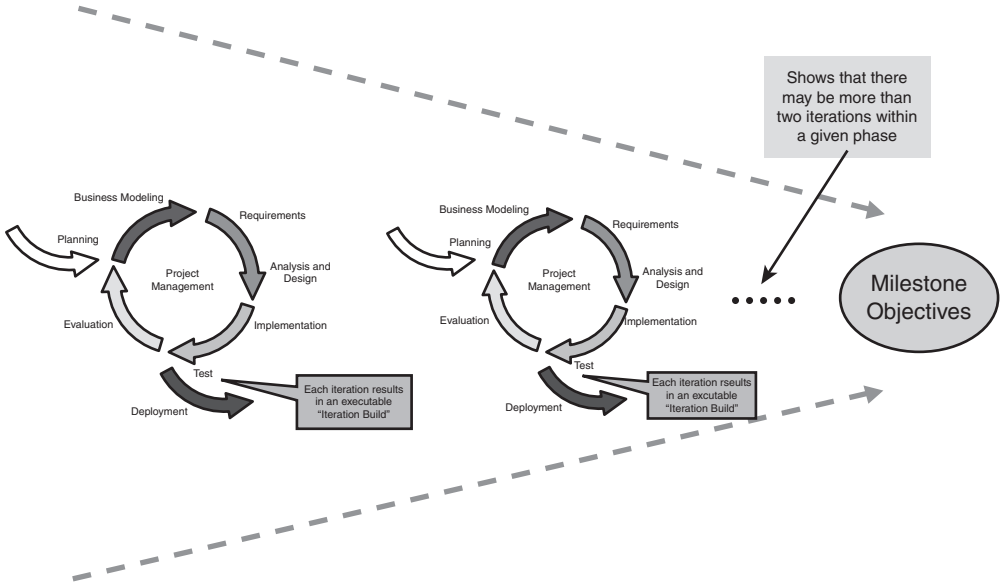


Figure 1-6 Iterations achieve objectives of phase milestone

Phases need to end with accomplishment milestones, as shown in Figure 1-5. Each milestone provides a critical decision point, or a go or no-go. Chapter 14, "Phases, Activities, and Milestones," discusses in detail all four phases and their respective objectives and evaluation criteria. Here, we will briefly review these phases. Please note that phases are executed in a sequence, as shown in Figure 1-5.

Inception Phase

The main goal of the Inception phase is to achieve concurrence among all stakeholders on the lifecycle objectives of the project. The following are the primary Inception phase objectives:

- To establish the project's scope and boundary conditions
- To identify the critical use cases of the system
- To exhibit and demonstrate one candidate architecture
- To estimate the overall cost and schedule for the project
- To produce detailed estimates for the Elaboration phase
- To estimate the potential risks
- To prepare the support environment for the project

The RUP is risk driven; the highest risks are identified earliest, and efforts are made to mitigate or address those risks as early in the project lifecycle as possible instead of pushing them forward. The Inception phase plays the most critical role in the project and will result in the first release of the product. In such cases, significant business and requirements risks need to be carefully managed. Accordingly, for new releases or enhancements of existing products, the Inception phase becomes much shorter. The Lifecycle Objectives milestone concludes the Inception phase. At that point, a major decision is made on whether to proceed with the project or cancel it.

Elaboration Phase

The main goal of the Elaboration phase is to baseline the architecture of the system to provide a stable basis for the bulk of the design and implementation effort in the Construction phase. The architecture evolves based on the most significant requirements and assessment of risks. To evaluate the stability of the architecture, one or more architectural prototypes may be developed. This architectural prototype is the executable architecture. The Elaboration phase objectives are as follows:

- To stabilize the architecture, requirements, and respective plans
- To sufficiently mitigate risks to predictably determine project cost and schedule
- To address all architecturally significant risks
- To establish a baselined architecture
- To produce an evolutionary prototype of production-quality components
- Optionally, to produce throw-away prototypes to mitigate specific risks such as design trade-offs, component reuse, and product feasibility

- To demonstrate that the baselined architecture will support the requirements of the system at a reasonable cost and in a reasonable time
- To establish a supportive environment

The Lifecycle Architecture milestone concludes the Elaboration phase, establishing a managed baseline for the architecture of the system and enabling the project team to scale during the Construction phase.

Construction Phase

The main goals of the Construction phase are to clarify the remaining requirements and complete the development of the system based on the baselined architecture. Construction phase objectives can be briefly summarized as follows:

- To minimize development costs through optimization of resource utilization by avoiding unnecessary scrap and rework and by achieving a degree of parallelism in the work of development teams
- To achieve adequate quality as rapidly as is practical
- To achieve useful executable versions (alpha, beta, and so on) as rapidly as practical
- To complete the analysis, design, development, and testing of all required functionality
- To iteratively and incrementally develop a complete product that is ready to transition to its user community
- To decide if the software, the sites, and the users are ready for the deployment of the solution

The Construction phase concludes with the Initial Operational Capability milestone, which determines whether the product is ready to be deployed into a beta-test environment.

Transition Phase

The overall goal of the Transition phase is to ensure that software is available for its users. It can span several iterations and includes testing the product in preparation for release and making minor adjustments based on user feedback. This feedback focuses primarily 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. Following are the primary objectives of the Transition phase:

- To validate the new system against user expectations (by beta testing)
- To train the end users and maintainers
- If applicable, to roll out the product to marketing, distribution, and sales teams
- To fine-tune the product by engaging in bug-fixing and creating performance and usability enhancements
- To conclude the assessment of the deployment baseline against the complete vision and the acceptance criteria for the product

- To achieve user self-supportability
- To achieve stakeholder concurrence that deployment baselines are complete and are consistent with the evaluation criteria of the vision

The Product Release milestone concludes this phase. A decision is made whether the objectives of the project were met.

RUP Phase Workflows

Each phase in RUP has a workflow, which describes the sequence in which activities from across various disciplines can be performed to achieve the objectives of the respective phase milestone. Chapter 14 explores in detail phase workflows and other process elements.

RUP Phases versus Waterfall Phases

RUP phases differ from traditional waterfall SDLC phases. In most cases, those who have been using the waterfall process equate RUP phases to traditional waterfall phases of Requirements, Analysis and Design, Implementation, and so on. The most common expression I hear when training software development teams on RUP is, " ... so it means that Inception is really about understanding and gathering requirements, Elaboration is really about architecting and designing, Construction about coding, and Transition about testing." I hear similar comments when I'm trying to help organizations adopt the RUP. The fact is that the phases in the RUP do not equate to those in the waterfall lifecycle. As discussed earlier, depending on which phase you are in, activities will be performed across multiple disciplines. The key differences can be summarized as in Table 1-1.

Table 1-1 Waterfall Phases versus RUP Phases

Waterfall Phase Characteristics	RUP Phase Characteristics
In any given phase, the activities are performed from a single area of concern. For example, during the Requirements phase, all activities related to requirements gathering and analysis are performed. No code is produced and no testing is carried out.	In any given RUP phase, depending on which phase it is, there will be activities from across multiple disciplines. For example, during the Elaboration phase, activities performed normally span all the core disciplines, including Requirements, Analysis and Design, Implementation, Test, and others.
Not all phases result in an executable deliverable. In fact, only Implementation and Test phases may produce executable deliverables.	With the exception of early Inception iterations, each iteration within each phase produces an executable deliverable.
A given waterfall phase employs a subset of team members who are skilled to perform related activities. This might lead to less than optimal resource utilization.	Producing an executable deliverable at the end of most iterations within RUP phases requires activities from across multiple disciplines to be performed and therefore engages the entire team.
Most waterfall phases result in document-based deliverables.	Most iterations within RUP phases result in an executable deliverable.

The later section "Iteration Maturity Levels" discusses different iteration patterns that have worked well for certain organizations.

Discipline

This section covers the important aspects of disciplines in the RUP. However, before we get into all the RUP details, let's see what the term *discipline* means and how and why it is one of the core components in the RUP.

Meaning of Discipline

According to the Merriam-Webster dictionary, the term **discipline** is defined as follows:

- From Latin *disciplina* teaching, learning, from *discipulus*
- To bring (a group) under control
- A field of study
- A rule or system of rules governing an activity

As you can see, the term *discipline* has been historically used in relation to learning, teaching, controlling, and governing. Discipline is also defined as a controlled behavior expected to produce a specific improvement. Furthermore, it is defined as a pattern of behavior made up of a set of rules and methods. The next section demonstrates how most of these definitions of *discipline* apply to RUP in one form or the other.

Briefly, in RUP, a discipline is defined as a categorization of activities based on similarity of concerns and cooperation of work effort. A discipline is a collection of activities that are related to a major "area of concern" (or "a field of study," as discussed earlier) within the overall project. In RUP, an **activity** is a process element that supports the nesting and logical grouping of related process elements, such as a descriptor[2] and subactivities, thus forming breakdown structures. The grouping of activities into disciplines is mainly an aid to understanding the project from a traditional waterfall perspective; that is, in a traditional waterfall project, your phases are called Requirements, Analysis, Design, Implementation, Testing, and so on. Therefore, within a waterfall project, you focus on a single discipline and associated artifacts for that discipline. For instance, when you have finished the Requirements phase of a waterfall project, you will gain final approval from the customer and move on to the next phase which, in most cases, is Analysis. In the RUP, although it is more common to perform activities concurrently across several disciplines at any given point during the life of a project (for example, certain Requirements activities are performed in close coordination with Analysis and Design activities), separating these activities into distinct disciplines is simply an effective way to organize content, which makes comprehension and learning easier. In addition, because the skill-sets needed to perform the tasks in one area of concern are probably similar, logical grouping of these activities simplifies the way different roles are organized. This enables us to align a small set of roles along discipline lines.

[2] According to the RUP, "A Descriptor is a Process element that represents a Method Content Element in the Process. The Descriptor provides the ability to override or add to what is in the original Method Content Element. Descriptors include Role, Task, and Work Product Descriptors."

The Role of Disciplines in the Software Engineering Process and the RUP

According to one of the historical definitions, the term *discipline* is a field of study that allows us to learn about that field in detail. Software engineering can be considered one of the many disciplines of engineering. In the RUP, however, a discipline refers to a specific area of concern (or a field of study, as mentioned earlier) within software engineering. For instance, Analysis and Design is one of the disciplines in RUP, which is itself a field of study and requires dedicated learning and distinct skill-sets. In addition, disciplines in RUP allow you to govern the activities you perform within that discipline. A discipline in RUP gives you all the guidance you require to learn not only when to perform a given activity but also how to perform it. Therefore, disciplines in RUP allow you to bring closely related activities under control.

We will see in detail how these related activities are governed and performed in an organized manner, not in isolation and not haphazardly. In fact, a recommended sequence should be followed to achieve optimal performance and maximize productivity and predictability. Note that although disciplines propose a recommended sequence of activities, these are truly performed in parallel with activities from other relevant (based on where you are in the project lifecycle) disciplines. When I was engaged in enabling one of the financial institutions to adopt RUP, I had to have separate sessions and workshops with System Analysts, Business Analysts, Project Managers, and others. During those sessions, the discipline workflows really proved helpful from the perspective of the involvement of a given role and the related activities to be performed across the RUP lifecycle. The workflows helped the people with given roles appreciate the effort that was required within their discipline.

A clear understanding of these relationships between roles and disciplines and the appreciation of how different roles from across different disciplines collaborate throughout the project lifecycle is important. It is crucial that you establish a clear understanding of this concept right from the beginning, and it will be helpful as you become immersed in the iterative development world. To ensure that you understand this well, Figure 1-7 shows the activities that are performed during an iteration within an Inception phase. Please look carefully at the activities in this figure and then compare it to the hump chart shown in Figure 1-4. You will be able to appreciate the relationship between the height of humps and the activities as they are aligned for each discipline.

The benefits provided by separating the RUP activities into various disciplines are summarized as follows.

- Makes the activities easier to comprehend.
- Proves useful when customizing a given discipline to meet the specific needs of the project or when defining a set of organizational standard processes. For a detailed discussion on RUP customizations and tailoring, please refer to Part IV of this book, "Tailoring and Tooling."
- Enables different roles to better and more effectively appreciate their responsibilities (in terms of the tasks/activities that they are responsible for) on a given project.
- Allows Project Managers to more effectively monitor and control these activities.

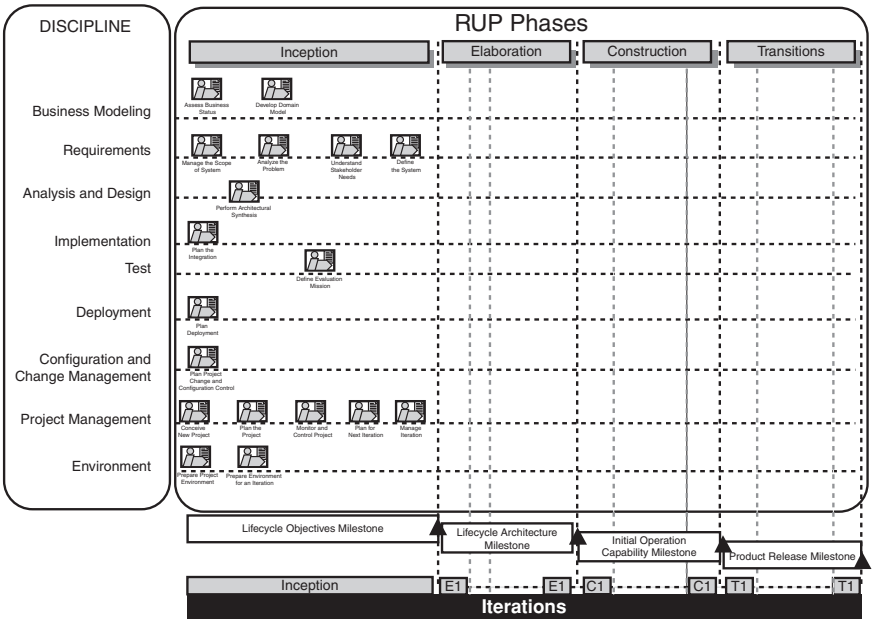


Figure 1-7 Inception iteration activities

Therefore, a discipline in RUP is a collection of activities that are related to a major area of concern or field of study. Each activity is further decomposed into subactivities or one or many tasks. Tasks require an input artifact or artifacts for their successful execution, and these in turn produce or refine some form of output artifact(s). Note that these artifacts can include both document-based artifacts and executables. Each task has an associated role (or roles) responsible for performing that task. To provide additional support and guidance, each discipline in the base RUP offers a set of standard template artifacts related to that discipline. These artifacts, as well as the process, can be (and should be) customized/tailored for a given project or organization.

Discipline Workflow

RUP models the *when* as workflows, and each discipline in RUP has a workflow. Like other workflows, a discipline's workflow is a semi-ordered sequence of activities performed by specific roles to achieve a particular goal. This semi-ordered nature of discipline workflows emphasizes that they cannot present the nuances of scheduling "real work," because they cannot depict the optionality of activities or iterative nature of real projects. Yet, they still have value as a way for us to understand the process by breaking it into smaller areas of concerns.

Keep in mind that the RUP framework, which these workflows are part of, constitutes guidance on a rich set of software engineering principles. It is applicable to projects of different size and complexity, as well as to different development environments and domains. This means that

no single project or organization will benefit from using all of RUP. Applying all of RUP will likely result in an inefficient project environment, where teams will struggle to keep focused on the important tasks and struggle to find the right set of information. Thus, as discussed earlier in this chapter, it is recommended that RUP be tailored to provide an appropriate and customized process for developing software. RUP tailoring and related tools are discussed in greater detail in Part IV of this book.

It is important to understand that the sequence of activities in each of the workflows is based on best practices. It should not, by any stretch of the imagination, be taken as a mandatory sequence. As an important component of tailoring the RUP framework, these workflows should be customized to suit project or organizational needs. This customization might require redefining some of these sequences.

Discipline Work Breakdown Structure

According to the Project Management Institute (PMI), the project's work breakdown structure (WBS) provides the relationship among *all* the components of the project and the project deliverables. WBS, according to PMI, is a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. It organizes and defines the total scope of the project. Each descending level represents an increasingly detailed definition of the project work.

You will see that the RUP adopts a slightly different view of WBS. Discipline WBS in RUP represents the *activities*-oriented hierarchical decomposition of the project effort specific to the respective discipline. Each descending level represents an increasingly detailed definition of the project work. In the RUP, the WBS provides *mostly*[3] four descending level of details. These levels include Discipline, Activity, Sub-Activity/Task, and Step. Activity is a process element that supports the nesting and logical grouping of related process elements such as descriptor and subactivities, thus forming breakdown structures. Task is a unit of work that a role may be asked to perform. Step is a content element used to organize tasks into parts or subunits of work. Note that it is at the Task level that RUP associates the roles and the artifacts produced, modified, or used. These levels are expressed visually in Figure 1-8.

Role

In RUP, a **role** is a definition of the behavior and responsibilities of an individual, or a set of individuals working together as a team, within the context of a business organization. RUP uses the concept of role to model the *who* of the software engineering process. This describes a role played by an individual or team within the project. Each role may be realized by many individuals or teams, and each individual or team may perform many different roles. For instance, Project Manager and Process Engineer are two different roles defined in RUP. On a smaller project, these two

[3] Sometimes you will see additional levels of decomposition. For instance, in some cases, activities are further decomposed into activities, which in turn are decomposed into tasks. The business modeling discipline is a good example in which a few activities have their own activity model.

roles may be performed by a single individual. On a larger project, there might be more than one individual performing the Project Manager and Process Engineer roles. The important point to note here is that whoever is performing any given role needs to have the right skill-set to perform the activities defined in RUP. We will see what primary roles are associated with a given discipline and for which they are primarily responsible. As shown in Figure 1-8, a role performs a task.

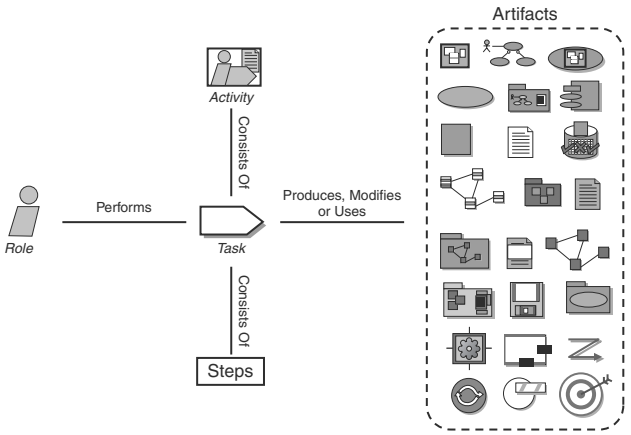


Figure 1-8 Level of activity decomposition

These individuals or teams adopt specific roles when they perform certain activities; therefore, for any activity, RUP can tell us the roles that participate in that activity. Activities may be broken down into finer levels of detail as needed. RUP also provides complete guidance and best practice details on *how* to perform each activity and task.

Discipline Artifacts

Related to each activity are the artifacts, which are either produced or refined depending on when the activity is performed during the project lifecycle. An **artifact** is a work product that is produced, modified, or used by a task and defines an area of responsibility. For any nontrivial development effort, especially where large development teams are involved, the artifacts are most likely to be subject to version control and configuration management. In the RUP, artifacts are generally not paper documents. Note that Figure 1-8 (just for illustrative purposes) shows just a subset of artifacts. Therefore, artifacts are inputs and outputs to the activities performed throughout the project lifecycle; they may be source code, executable programs, standards, documentation, and so on. Part III, "Rational Unified Process: Content and Process Elements," discusses key artifacts of RUP disciplines in detail.

The Hump Chart—Putting Phases, Iterations, Milestones, and Disciplines Together

The reality is that a mini-waterfall project exists within each iteration of the RUP project.

Having discussed phases, iterations, milestones, disciplines, and other key concepts, let's revisit the famous RUP hump diagram shown in Figure 1-4 to put these together and discuss the interrelationships in more detail.

The horizontal axis represents iterations and the progress of a RUP lifecycle. As discussed, every RUP project is divided into four significant phases called Inception, Elaboration, Construction, and Transition. We will discuss the four RUP phases in greater detail later in this book. The dashed lines between the phases are called **milestones**, which mark checkpoints in RUP. These milestones present a go/no-go decision by project management when artifacts have reached a specified state. The word *sign-off* or *freeze* does not exist for RUP artifacts, but artifacts need to reach specific states depending on the time in the RUP project lifecycle reflecting the level of their maturity. For example, the first draft risk list is developed during the Inception phase and is refined during the entire project lifecycle.

The vertical axis, called **disciplines** (called **workflows** in earlier versions of RUP), defines the activities performed during an IT project. The RUP has nine disciplines. Six of them are directly linked to software engineering activities and are also known as **core** disciplines. These are as follows:

- Business Modeling
- Requirements
- Analysis and Design
- Implementation
- Test
- Deployment

The other three are also called **umbrella** activities (also known as **supporting** disciplines), because they are concerned with the overall management and structure of a RUP project:

- Configuration and Change Management
- Project Management
- Environment

We will discuss these disciplines in greater details in Part III of this book.

Iteration Maturity Levels

So far, we have discussed the basics of the RUP. With this brief introduction, let's try to develop a clearer understanding of iteration design evolution. Our goal in the following discussion will be to gain insights into how industries have implemented the RUP and iterative development. We will discuss how different companies have adapted and adopted the RUP and have gone through a true

lifecycle of successfully evolving and institutionalizing the new methodology. You will notice how the returns in incremental and iterative development investments increase with the increasing level of organizational maturity. We will also explore how the design of iterations evolved as organizations matured over time. Before we get into all these details, note that, at its core, the focus shifts as we progress through the project lifecycle. This is demonstrated in Figure 1-9.

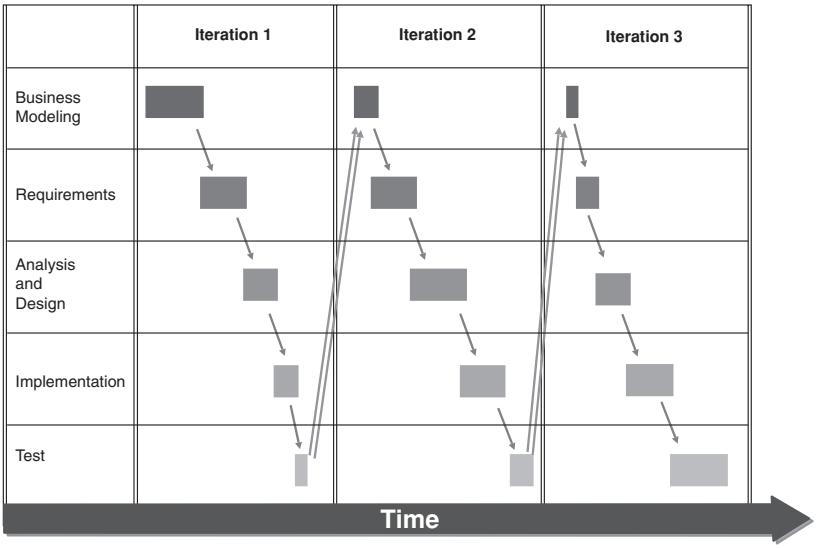


Figure 1-9 Changing focus across the RUP lifecycle (from RUP material)

The relative size of a given box aligned with iteration and discipline respectively shows the level of focus or effort. For instance, during Iteration 1, the primary focus is on Business Modeling, whereas during Iteration 3, the primary focus is on Test activities. Regardless of how iterations are designed, the focus potentially stays somewhat consistent with this figure. With this in mind, let us now look at the iteration maturity lifecycle pattern that I have most frequently encountered in my years of consulting and related professional experiences. As we discuss this, please keep in mind that where an organization starts depends on a number of factors, including organization size, culture, complexity, iterative development experience, and structure. Organizations evolve their processes as they mature and as teams become more experienced with agile-like iterative and incremental development methodologies. I will refer to these different maturity levels as *iteration maturity levels*.

Iteration Maturity Level 1—Incremental Mini-Waterfall

How do you change culture? The change cannot happen overnight, especially if you are dealing with global organizations that span multiple cultures. In addition to project-specific complexities, we need to take into account other factors such as organizational size and differences in execution

models. Cultural change requires behavior change, and behavior change is driven by evolving organizational processes. Sponsorship from the senior management becomes absolutely critical. Over time, the integral of all these small incremental process changes impact the team's behavior and enables the team to embrace the new culture and achieve the desired results.

Cultural change is not the topic of discussion here. However, it plays an important role in the way software development organizations are able to enhance their productivity and improve product quality by adapting and adopting the RUP. Enabling large, mature waterfall-based software development organizations to embrace a somewhat revolutionary approach is challenging. In situations like these, such organizations need to take baby steps, demonstrate value, embrace and institutionalize processes, and proceed forward. Big bang may work, but with potentially large disruption.

"Mini-waterfall" may be that first approach that you would like such organizations to take. Get them to think about producing incremental builds and executables as early in the project lifecycle as possible. Don't underestimate the challenges surrounding controls and checks even in the iterative and incremental development world. It will require well-structured management. This is especially true for those organizations that are regulated heavily by industry and other government bodies. Such organizations need to ensure compliance with those regulatory requirements, perhaps by introducing controls throughout the project lifecycle.

The need for such controls is one reason that waterfall development found its way into these organizations and is still there. Waterfall development clearly separates discipline-based phases, which enables management to review and provide the necessary approvals before projects are allowed to proceed further. For example, the Requirements document needs to be reviewed, approved, and frozen prior to analysis, design, or implementation efforts. Such control requirements are best achieved through waterfall development. Some organizations even have their SDLC (Software Development Lifecycle) team report into the compliance department. Both end up with conflicting goals—SDLC wants efficiencies, whereas compliance requires maximum controls. When the waterfall approach is completely embedded in organizational cultures, incremental adoption might be a reasonable option. Let's see what we mean by this mini-waterfall like approach.

Figure 1-10 presents a unique but supporting perspective to the RUP hump diagram shown in Figure 1-4. Let's see how.

Figure 1-10 takes the RUP process framework and presents the way it might look like if it is applied to a green-field type of project (custom development from the ground up). For the sake of our discussion and simplicity, we will only look at a subset of RUP disciplines and will assume that the project consists of six iterations named I1 through I6. Dark gray, light gray, and white represent the focus levels. The figure presents a simplistic view of the way iterations are executed, the relationships with other iterations, and the phases.

Note that each iteration shown in Figure 1-10 resembles a mini-waterfall project. What does that mean? It means that, depending on the goals and objectives of a given iteration and

where in the RUP lifecycle it is taking place, the degree of effort (also termed as **focus**) on activities from across different disciplines will shift. As you progress through the lifecycle represented in Figure 1-10, the focus shifts from being analysis driven in the early iterations to implementation, testing, and deployment in the later ones. This change of focus happens primarily because of knowledge gained about the business and the problem. Later iterations might be characterized by few refinements to the business model and requirements and greater focus on implementation and testing. This change in focus across the life of a given project is driven primarily by effectively and efficiently managing risks. Each iteration converges on project goals.

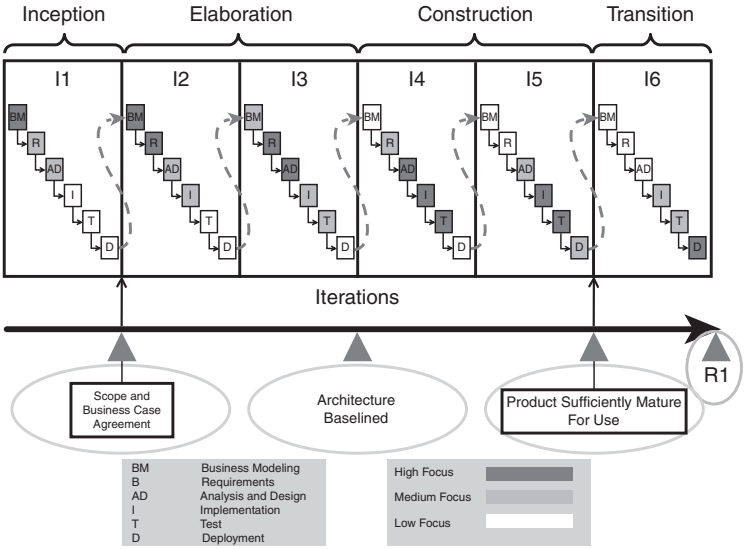


Figure 1-10 Iterative-incremental development and the Rational Unified Process

Let's look at the specific case represented in Figure 1-10. The Inception phase consists of only one iteration: I1. Because it is the first iteration, the focus is on performing Business Modeling to gain enough understanding of the business domain to proceed further. Note, however, the medium effort being invested in the Requirements and Analysis and Design disciplines and the low effort made in the Test and Deployment disciplines.

It works like this. Although you might be learning more of the business domain and associated business processes, it is important to understand the stakeholder needs by having requirements workshops and producing a use-case model and supplementary specs while analyzing the problem. You can also develop a prototype, depending on your understanding, and strategize about testing and deployment. Earliest iterations might not require executable deliverables. However, each iteration should have specific goals that are evaluated at the end.

In iterations I2 and I3, the focus shifts to more designing, programming, and testing while refining the vision and the environment. Iterations I4 and I5 focus on programming and testing, with minor requirements changes. As shown in Figure 1-10, iteration I6 focuses on beta testing, doing some final programming and documentation, and deployment. One element that remains consistent throughout the RUP lifecycle is that each iteration takes the project closer to its stated goals.

By now you should understand that in this specific iteration pattern, iterations are more like mini-waterfall projects, each with its own goal(s), which ultimately help to get the overall project closer to meeting its goals. The next big question that senior managers, who are more concerned with financial aspects of the project, have is about gaining some control over the life of a project and not losing control from iteration to iteration. This is known as **time-boxing**. Few companies would allow you to continue iterations and refinement and never close the project. Any statement similar to "A project plan is continuingly evolved throughout the project lifecycle" is hard to sell to senior managers.

To address these and similar challenges, we need to do time-boxing at the project, phase, and iteration levels. As you continue through the RUP lifecycle, from one iteration to the next, note the four major milestones, as mentioned earlier:

- **Lifecycle Objectives milestone**—Scope and business case agreed
- **Lifecycle Architecture milestone**—Architecture baselined
- **Initial Operational Capability milestone**—Product sufficiently mature for use
- **Product Release milestone**—Product release

Based on the complexity and size of the project, an iteration can be anywhere from 2 to 6 weeks long, and respective phases can be composed of different numbers of iterations. Iterations are logically grouped to meet key milestones at the end of each phase. These milestones mark the accomplishment of clearly specified phase objectives.

Iteration Maturity Level 2—Incremental Mini-Waterfall with Feedback Loops

After firms perform at the mini-waterfall level, the next level is to further refine the iteration design such that feedback loops exist. These feedback loops are shown in Figure 1-11.

Such feedback loops enable the continuous evolution of not only the iteration builds but also the continuous refinement of the related artifacts. This builds traceability and consistency across all artifacts starting with the vision and continuing to the executable code. More and more tools are being developed to support such a model.

Iteration Maturity Level 3—Optimizing Iterative and Incremental Development

At this level, organizations are not only building higher quality products but are also optimizing their resource utilization. As shown in Figure 1-12, at any given point during an iteration at this maturity level, the following steps might be taking place.

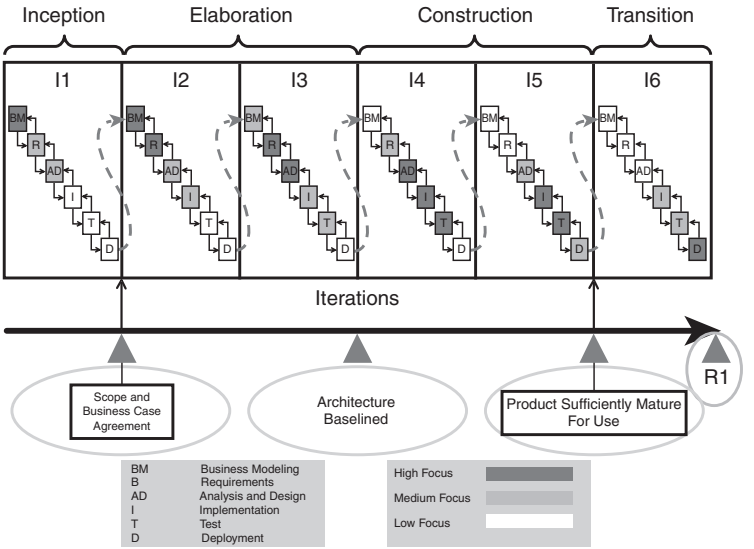


Figure 1-11 Incremental mini-waterfall with feedback loops

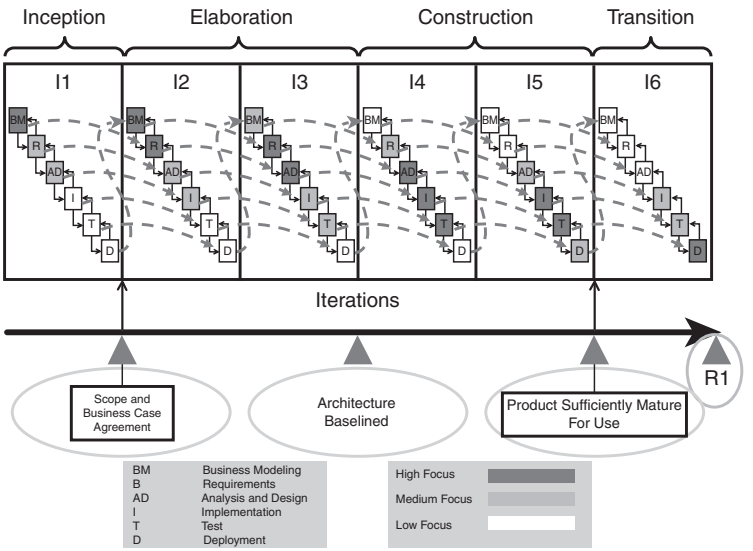


Figure 1-12 Optimizing iterative and incremental development

1. Artifacts that are required by activities/tasks within the same iteration from across other disciplines are produced/refined. For example, architecturally significant use cases need to be identified before the related Analysis and Design activities can be performed, code is required before it can be tested, and so on.
2. Artifacts that are required by activities/tasks within the following iteration might be produced/refined.
3. Artifacts might be refined based on feedback.

Greater efficiencies and higher productivity as a result of such concurrent and iterative and incremental development will most certainly require strong management, a mature and experienced team, and a well-integrated tool suite.

Evolution of the Rational Unified Process

Now that we have a clearer understanding of the RUP architecture, let's discuss how the RUP evolved over the years. RUP often comes across as an overloaded term. For example, many believe that the IBM Rational Unified Process is inseparable from the Rational Unified Process product. Others think that the IBM Rational Unified Process requires other IBM Rational software to function. This section briefly discusses the evolution of RUP and will try to separate myths from realities.

The first version of RUP was released in 1998, but it was heavily influenced by its predecessor, known as **Objectory**™, which dates back to 1988 (see Figure 1-13). With its long history, RUP as an iterative development process has a proven track record unparalleled in the IT industry.

No other modern software engineering process has a higher adoption or success rate than RUP, which is attracting more and more organizations. The RUP story has continued, especially after IBM Rational announced the accomplishment of another milestone in October 2005, which fundamentally changed the distribution, configuration, and deployment of RUP.

This is not a RUP history book, and this history is not important for your certification. However, a brief discussion of the most recent RUP-related decisions will help you understand where the RUP journey might go. This may, in fact, further inspire you to achieve the RUP certification.

In October 2005, IBM Rational donated a subset of the RUP process framework, now known as the Basic Unified Process (BUP),[4] to the Eclipse Foundation. This framework was modified as part of the Eclipse Process Framework project, and the resulting extensible process is named the Open Unified Process, or OpenUP. The distribution of a subset of RUP through an Eclipse project will allow all interested parties to adopt the concepts of RUP as an open-source process framework. This donation will also encourage software engineers to use BUP and develop open-source process enhancements for RUP.

[4] http://www.eclipse.org/proposals/beacon/

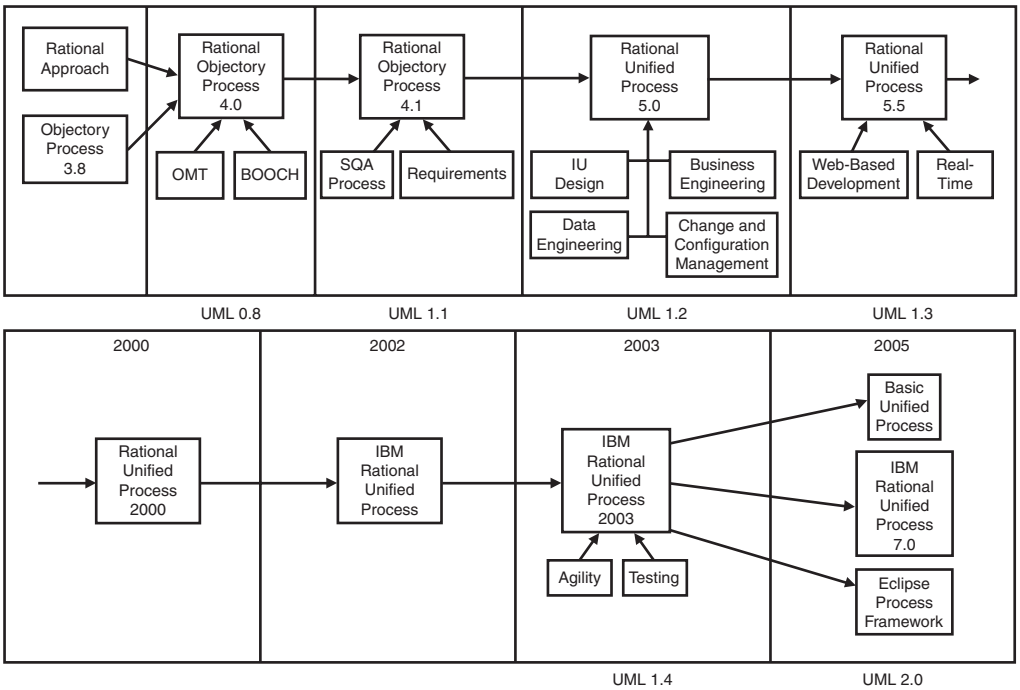


Figure 1-13 RUP evolution

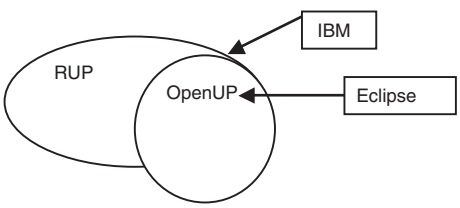


Figure 1-14 OpenUP content

Another significant milestone is the development of the IBM Rational Method Composer[5] (RMC), the new Eclipse-based product to configure and distribute customized processes like OpenUP or RUP. This product will supersede the previous IBM Rational proprietary RUP tools (Rational Workbench, RUP Modeler, and RUP Organizer), which were used to customize RUP. As with the process donation, the basic capabilities of RMC were donated to Eclipse; the resulting tool, Eclipse Process Framework composer, is available free as an open source application.

[5] http://www-306.ibm.com/software/awdtools/rmc/

Figure 1-15 shows that the BUP knowledgebase is now in pieces available as open source, whereas the IBM RMC is a tool for authoring and publishing BUP or RUP. The clear separation between the framework and tool will encourage a distribution of the concepts of iterative software engineering in the industry, whereas the tool will help process engineers to tailor it.

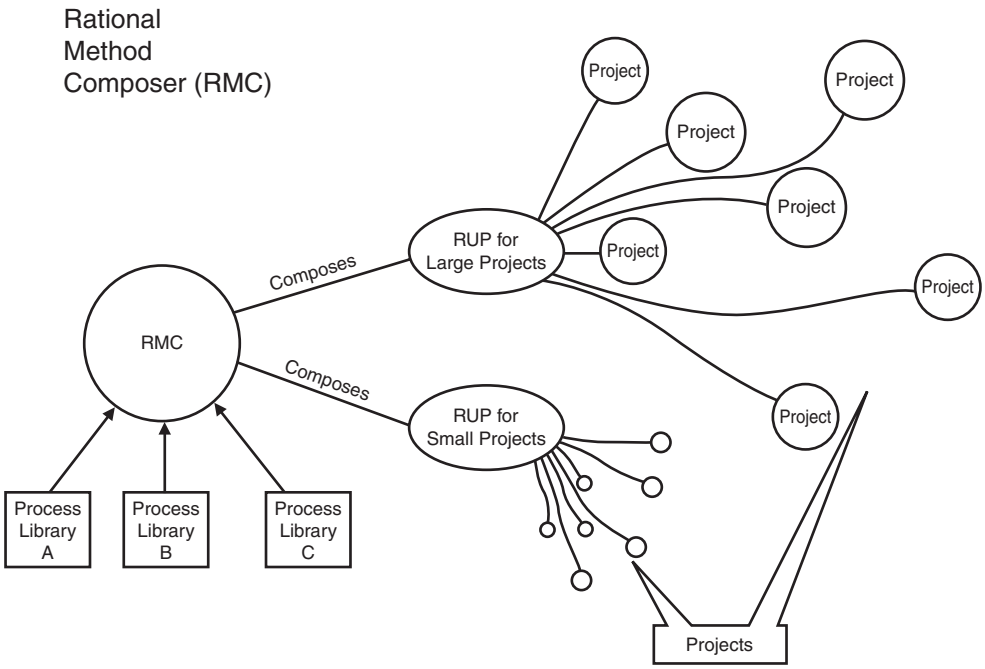


Figure 1-15 RMC and RUP

Why would IBM Rational take such a step? IBM originated the Eclipse foundation and promoted open source development for years. It is the logical next step to do the same thing with a software engineering process, which ties all the concepts under one umbrella. Although IBM has gathered and organized the process for more than a decade, the donation to Eclipse will spread the message through the open source community with one goal in mind: improving the industry's IT processes.

IBM Rational as a division inside IBM delivered software engineering tools to the IT industry for many years. Focusing on the RMC and the integration with the IBM Rational Portfolio Manager (RPM) and promoting the open source process framework will strengthen the role of IBM Rational in this space while the industry benefits from the donation.

Challenges in Identifying RUP Solution Designers

Implementing RUP requires dedication, determination, sponsorship, and expertise. Related to expertise, if you need a RUP Solution Designer, you will appreciate the significant challenge of finding a RUP expert. It is not easy to identify a RUP expert. RUP encompasses a range of disciplines (both Core and Supporting), each of which has a specialized and a dedicated field of study. For instance, there is a Project Management discipline, which focuses on project planning, risk management, monitoring progress, and metrics. Institutions such as the Project Management Institute (PMI) are dedicated to the growth of project management as a profession and provide related certifications such as the Project Management Professional (PMP). Can we then say that a PMP can be a good RUP Project Manager? Probably! However, this person will certainly need to have experienced the iterative and incremental development supported by the RUP. Otherwise, this might turn into a project in itself to mentor a PMP to become a RUP Project Manager. We have seen organizations failing to implement the RUP process due to lack of their Project Managers' iterative and incremental development experience. In short, professionals who aspire to become strong RUP Solution Designers need to have a range of technical and managerial skills to succeed. Such individuals are not easy to find. And when you indeed think you have found them, it is hard to gauge their depth and breadth of knowledge.

You will be able to appreciate the uniqueness of the RUP Solution Designer Certification and the importance of achieving it if you really have such a rare skill-set. With the RUP Solution Designer Certification, your name will be among those few professionals who have been and will be in great demand. With this certification, software engineers and managers can now register after successful completion at http://www-304.ibm.com/jct09001d/member.nsf.

Past, Present, and Future of RUP Certification

In the past, when IBM Rational owned the process and the certification process, individuals were advised to visit the two-day training course PRJ270, Essentials of Rational Unified Process version 2, directly from IBM Rational. Originally thought of as an official measure for future RUP instructors, the certification followed the content of the course. The course was not a prerequisite to achieve certification (and it still is not), but the student book and the course content contained valuable information for preparing for the certification exam. Over time and because of the acquisition of Rational by IBM, the RUP certification came to be seen from a different perspective. Being part of a series of official technical certification exams, the RUP certification became exposed to a much broader audience.

Our goal was to compile a resource to allow RUP professionals to not only use RUP in their daily work as a reference, but also to gain the knowledge required to successfully achieve the RUP Certification. This book, combined with the IBM Rational RUP course (the aforementioned Essentials of Rational Unified Process) and real-world experience will enable you to achieve your RUP Solution Designer Certification, hopefully in the first attempt.

With the increasing awareness and adoption of iterative-incremental software engineering processes through the open source community, the RUP Solution Designer Certification will undoubtedly distinguish experts from casual users. This certification will not only give you the industry recognition of being a RUP Solution Designer, but you will also promote the process in a manner similar to that of an ambassador. This book does not replace the RUP training course by any means, and we do not guarantee that you will pass the exam on your first attempt. But if you follow the guidelines provided in this book, practice with the sample questions, and appropriately use the references to other RUP resources, you can prepare yourself for a successful examination. We dedicated the entirety of Part IV of our book to the actual examination and its logistics.

Summary

This chapter introduced some of the most critical components of the RUP process framework. It covered the history and evolution of RUP and its certification program from its origin until today. It also looked ahead into a possible future of RUP. Using some basic RUP terminologies, this chapter mapped RUP concepts to the structure of the book. Now that you have read this chapter, try to name a few key principles, disciplines, and RUP phases and approach the exercises. By the end of this book, you will have internalized them.

One key lesson for waterfall methodology practitioners is that compared to the waterfall approach, iterative-development can be seen as many mini-waterfalls. As is apparent from Figure 1-7, another important take-away is that activities from numerous disciplines are performed in parallel in any given iteration and not sequentially.

Sample Questions

The correct answers to these questions can be found in the Appendix, "Answers to Sample Questions."

1. In the Rational Unified Process, which of the following provides the means of assessing the progress of a project?
   a. Discipline
   b. Project Management
   c. Milestone
   d. Project Management tools

2. In the Rational Unified Process, which of the following contains activities from numerous areas of study?
   a. Iteration workflow
   b. Phase workflow
   c. Discipline workflow
   d. Project workflow

3. Which of the following is true about RUP disciplines?
   a. Enable the project manager to plan the project in a traditional waterfall fashion more effectively
   b. Represent activities that compose areas of concern in a project
   c. Correspond directly to RUP roles
   d. Are implemented one at a time in serial fashion

4. Which of the following is a key component of the RUP discipline? (Select all that apply.)
   a. Activity
   b. Role
   c. Artifact
   d. Phase

5. In the RUP, which of the following is true about the Work Breakdown Structure? (Select all that apply.)
   a. The RUP Work Breakdown Structure is a deliverable-oriented hierarchical decomposition of project effort.
   b. The RUP Work Breakdown Structure provides relationships among all the components of the project and project deliverables.
   c. The RUP Work Breakdown Structure is an activity-oriented hierarchical decomposition of the project effort.
   d. The RUP Work Breakdown Structure enables the project manager to more effectively plan the sequence in which activities should be performed.

6. Which of the following is a unit of work that a role may be asked to perform?
   a. Task
   b. Activity
   c. Workflow
   d. Work Unit

7. Which of the following is true about the relationship between the RUP Work Breakdown Structures and the RUP Workflows? (Select all that apply.)
   a. The RUP Work Breakdown Structure presents an activities-oriented hierarchical decomposition.
   b. The RUP Work Breakdown Structure presents a deliverable-oriented hierarchical decomposition.
   c. The RUP Work Breakdown Structure contains the tasks that are presented in RUP workflows.
   d. The RUP Work Breakdown Structure activities can be repeated across various RUP discipline workflows.

8. Which of the following is true about a workflow diagram for a discipline? (Select all that apply.)
   a. It shows the sequence in which activities should be performed.
   b. An activity within a workflow can consist of a workflow.
   c. A workflow is made up of tasks.
   d. RUP has more than one workflow for each discipline.

9. Which of the following is true about tasks? (Select all that apply.)
   a. One or more tasks can be carried out within any given activity.
   b. Tasks can be repeated across different activities within a discipline.
   c. Roles are not responsible for performing tasks. Instead, they are responsible for performing activities.
   d. Tasks produce, modify, or use an artifact.

10. Which of the following is true about a role? (Select all that apply.)
    a. A role is a definition of the behavior and responsibilities.
    b. An individual can assume a role.
    c. A role can be a set of individuals working together as a team.
    d. The role and the person performing it are interchangeable.

11. Separating the RUP activities into different disciplines provides which of the following key benefits? (Select all that apply.)
    a. Makes the activities easier to comprehend
    b. Proves useful when customizing a given discipline to meet the specific needs of the project
    c. Enables different roles to better and more effectively appreciate their responsibilities on a given project
    d. Allows project managers to more effectively monitor and control these activities

12. In RUP, a discipline is defined as which of the following?
    a. A categorization of tasks based on similarity of concerns and cooperation of work effort
    b. A categorization of activities based on similarity of concerns and cooperation of work effort
    c. A field of study
    d. A software engineering domain

13. Which of the following is used to describe in detail how to perform a particular activity?
    a. Template
    b. Guideline
    c. Checklist
    d. Concept

14. Which of the following are the key process elements in RUP? (Select all that apply.)
    a. Discipline
    b. Phase
    c. Actors
    d. Template
15. Which of the following is true of a discipline in RUP? (Select all that apply.)
    a. Gives you all the guidance you require to learn when to perform a given activity
    b. Should not be tailored because doing that affects the integrity of the discipline
    c Provides all the guidance you need to learn how to perform a given activity
    d. Produces documents
16. What important information does the RUP hump diagram present? (Select all that apply.)
    a. Illustrates the varying degrees of focus across different disciplines in the development and evolution of the solution across the project lifecycle
    b. Shows the relationship between RUP phases and RUP disciplines
    c. Shows the roles responsible for performing activities within a given discipline
    d. Provides guidance on the number of iterations to be managed within any given phase
17. Which of the following is a discipline in RUP? (Select all that apply.)
    a. Deployment
    b. Environment
    c. Development
    d. Production
18. Logical grouping of related activities into respective disciplines enables you to do which of the following? (Select all that apply.)
    a. Define the roles for each discipline.
    b. Structure the tasks performed by these roles.
    c. Define artifacts that are produced or refined by these roles when they perform the related activities.
    d. Define the sequence in which these activities should be performed.
19. An artifact is a formal work product that is produced, modified, or used by a task, defines an area of responsibility, and is subject to version control. Which of the following is true about artifacts? (Select all that apply.)
    a. An artifact can be a model, code, or a document.
    b. An artifact is a work product.
    c. An artifact can be a mandatory or optional input into an activity.
    d. An artifact can be output of an activity.

20. Which of the following enables you to assess the quality of a particular artifact?
    a. Checklist
    b. Checkpoint
    c. Guideline
    d. Template
21. Which of the following views is used in the Analysis and Design discipline?
    a. Deployment view
    b. Use-Case view
    c. Analysis view
    d. Physical view

References

IBM Rational Unified Process v7.0.

Perry, D., Wolf, A. (1992). Foundations for the study of software architecture, *ACM SIGSOFT Software Engineering Notes*, 17(4): 40–52.