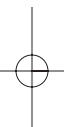



Chapter 4

An Introduction to the J2EE Developer Roadmap



In this chapter we introduce the J2EE Developer Roadmap. As we described in Chapter 3, An Introduction to the Rational Unified Process, RUP has been designed as a process framework from which customized processes can be developed. The J2EE Developer Roadmap is such a customization. The process chapters of this book (Chapters 6-9) provide a guided tour of the roadmap, and exemplify how it has been used to develop the sample J2EE Online Auction application described in Chapter 5, An Introduction to the Sample Application.

We begin this chapter by defining what a RUP roadmap is. We then describe what RUP activities and artifacts have been included in the J2EE Developer Roadmap, and provide a brief explanation as to why these particular process elements were selected. We end with an overview of the roadmap, demonstrating how the roadmap has been tailored to include specific J2EE process guidance.



What Is a RUP Roadmap?

RUP is a comprehensive and detailed process framework. It addresses almost every aspect of software development and can be intimidating to the first-time user. A question we often hear is “Where do I start?” RUP roadmaps provide the answer.

A RUP roadmap provides a starting point because it provides a tour through RUP with some viewpoint in mind. That viewpoint may be a specific



development context (for example, J2EE online enterprise systems) or a specific role (for example, software developer). A roadmap describes the elements of the process that are relevant to that viewpoint, and ignores the elements that are not relevant. A roadmap is intended to be a directed reading and learning aid and can be thought of as a customized view of the process.

J2EE Developer Roadmap—Scope and Rationale

The J2EE Developer Roadmap provides a customized view of RUP that has been tailored to meet the needs of the J2EE developer. It concentrates on providing guidance on the essential requirements, analysis and design, and implementation activities and artifacts from the developer's perspective. The RUP elements considered relevant to the J2EE developer are described in detail, while those considered less relevant are treated lightly or skipped altogether.

Before identifying the J2EE developer-relevant process elements, we should define what we mean by “developer.” We use the term “developer” to encompass multiple RUP-defined roles, which we describe in Table 4.1. All of these RUP roles correspond to one role in the J2EE platform specification—the *application component provider*. In other words, what we call J2EE developer in the book closely approximates to the application component provider role in the J2EE platform specification. The J2EE developer is responsible for taking a **Vision** of a system (possibly just a problem statement) through to an implementation of the system using the J2EE platform.

The J2EE Developer Roadmap includes activities from the Requirements, Analysis and Design, and Implementation disciplines of RUP and focuses on the development of the following models¹: **Use-Case Model**, **User-Experience Model**, **Design Model**, **Data Model**, **Implementation Model**, and **Deployment Model**. The software architecture is another important product of the development process that represents the architecturally significant aspects of these models. Software architecture is described in more detail in Appendix A, Describing a Software Architecture. We discuss other artifacts, but they are either contained within, or directly support the development of, one of these models.

Figure 4.1 provides an overview of these models and illustrates the relationships that exist between them. These models and their relationships are

¹ A *model* is a complete description of a system from a particular perspective (“complete” meaning you don’t need any additional information to understand the system from that perspective). A model contains a set of model elements. Two models cannot overlap.

Table 4.1 RUP Roles Included in the J2EE Developer Roadmap

<i>RUP Role</i>	<i>Brief Description</i>
Architecture Reviewer	The Architecture Reviewer plans and conducts the formal reviews of the overall software architecture.
Database Designer	The Database Designer defines the tables, indexes, views, constraints, triggers, stored procedures, tablespaces or storage parameters, and other database-specific constructs needed to store, retrieve, and delete persistent data.
Designer	The Designer defines the responsibilities, operations, attributes, and relationships of one or several design elements, and determines how they should be implemented.
Design Reviewer	The Design Reviewer plans and conducts the formal reviews of the design.
Implementer	The Implementer is responsible for implementing and testing design elements, in accordance with the project's adopted standards.
Implementation Reviewer	The Implementation Reviewer plans and conducts the formal reviews of the implementation.
Requirements Reviewer	The Requirements Reviewer plans and conducts the formal review of the requirements.
Requirements Specifier	The Requirements Specifier details the requirements.
Software Architect	The Software Architect leads and coordinates technical activities that establish the overall structure of the system, the key system elements, and the interfaces between these elements.
System Analyst	The System Analyst leads and coordinates requirements elicitation by outlining the system's functionality and by scoping the system.
User-Experience Designer ²	The User-Experience Designer defines the user actions, dynamic content, and navigation paths of one or several Screens, and determines how they should be implemented.
User-Experience Reviewer	The User-Experience Reviewer plans and conducts the formal reviews of the system's user experience.

described briefly in the section below, and will be discussed in detail within their respective process chapters.

The **Use-Case Model** describes the intended behavior of the system as seen through its interaction with its environment, and serves as a contract between the customer and the developers. The **Use-Case Model** is the foundation of the other models, as it contains a complete specification of the observable system behavior. It is used as an essential input to activities in analysis, design, and test.

² The user-experience process elements (for example, the **User-Experience Designer** role, the **User-Experience Reviewer** role, and the **User-Experience Model** artifact, which includes **Screens**, **Use-Case Storyboards**, and **Navigation Maps**) are not part of classic RUP, but are described in the User-Experience Modeling plug-in to RUP.

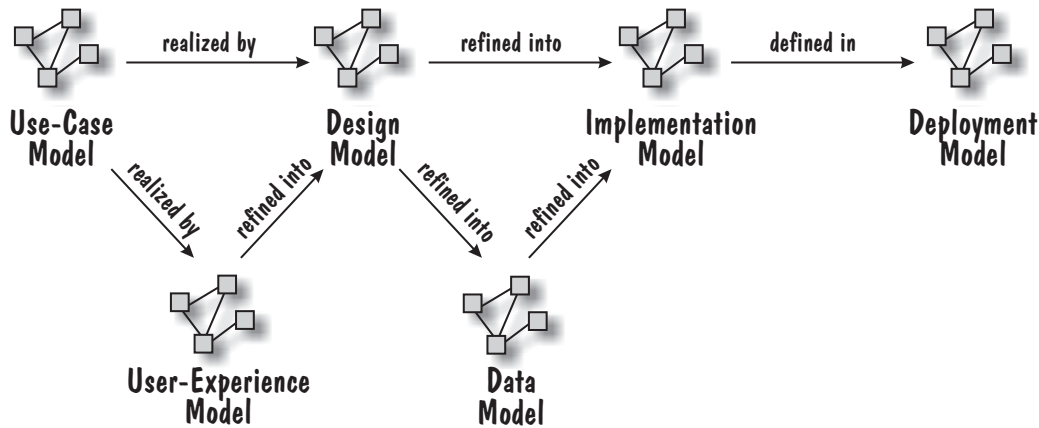


Figure 4.1 J2EE Roadmap System Models

The **User-Experience Model** describes what the user will see when interacting with the system. It describes the **Screens**, the dynamic content that appears on the **Screens**, and how the user navigates through the **Screens** to perform the system **Use Cases**. It provides the contract between the presentation and the business elements of the system and describes how the content provided by the business elements is presented to the user.

The **Design Model** describes the realization of the system **Use Cases**, and serves as an abstraction of the **Implementation Model**. It is a comprehensive, composite artifact encompassing **Analysis Classes**³, **Design Classes**, **Framework Components**, **Design Subsystems**, **Design Packages**, and **Use-Case Realizations**. The **Design Model** is as an essential input to the implementation activities.

The **Data Model** describes the persistent data in the system, including any behavior defined in the database, such as stored procedures, triggers, constraints, and so forth. The **Data Model** contains a set of model elements, which represent the physical storage of the persistent **Design Model** elements. It defines the mapping between the persistent **Design Classes** and the persistent data structures, and also defines the persistent data structures themselves. The **Data Model** is needed when the persistent data structures cannot be inferred from the structure of persistent classes in the **Design Model** and a mapping from the **Design Model** to the persistence storage mechanism must be explicitly defined.

³ In the J2EE Development Roadmap, **Analysis Classes** are initially created in the **Design Model**, and then evolve into design elements. An analysis view of the system is not maintained in a separate **Analysis Model**.

The **Implementation Model** is a UML model of the implementation of the **Design Model** and **Data Model** elements. It describes the **Implementation Directories** and **Implementation Files** needed to build and manage the system in the development environment. These files include both the operational files, such as executables, deployment descriptors, and so on, as well as the source code files from which the operational files are derived. The **Implementation Model** can be a very useful visualization of the relationships (for example, traceability) between the **Implementation Model** elements that represent the implementation and the elements in the other system models. Given the right toolset, the consistency between the visual representation of the **Implementation Model** and the physical files of the **Implementation Model** can be maintained using roundtrip engineering.

The **Deployment Model** shows the configuration of processing nodes at runtime, the communication links between them, and the **Implementation Model** elements that are deployed on them. It describes the distribution of behavior across nodes.

When deciding on the scope J2EE Developer Roadmap, our goal was to only include those artifacts that are directly relevant to the J2EE developer. It was our intention that the selected artifacts would serve as “stepping stones” from the **Vision** of the system to its implementation. Once the artifacts were selected, we only included activities that supported the production of these artifacts. The results are shown in Table 4.2, which summarizes the RUP activities and artifacts included in the J2EE Developer Roadmap. The table is organized by book chapter and is meant to provide an overview of the roadmap. The artifacts listed in parentheses represent “subartifacts” of the artifact preceding the parentheses (subartifacts are artifacts that are contained within another artifact). Some of the artifacts are referenced in the roadmap, but their development is considered outside of its scope. These artifacts are considered to be provided to the roadmap and are indicated in the table with “[provided]” following their name.

Some may question the inclusion of the requirements activities and artifacts in a process that is focused on the developer. We have chosen to include them since the requirements artifacts, especially the **Use Cases**, drive all other aspects of the process.

The fact that we excluded some of the RUP process elements from the J2EE Developer Roadmap does not mean that those elements are not important. They

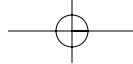
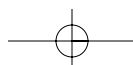


Table 4.2 J2EE Developer Roadmap Activities and Artifacts

<i>Chapter</i>	<i>Workflow Detail</i>	<i>Activities</i>	<i>Artifacts</i>
6—Requirements	Define the System Refine the System Definition	<ul style="list-style-type: none"> ◆ Capture a Common Vocabulary ◆ Find Actors and Use Cases ◆ Prioritize Use Cases ◆ Review the Requirements ◆ Detail a Use Case ◆ Structure the Use-Case Model ◆ Review the Requirements 	<ul style="list-style-type: none"> ◆ Change Request⁴ ◆ Glossary ◆ Iteration Plan [provided] ◆ Risk List [provided] ◆ Review Record ◆ Supplementary Specification ◆ Use-Case Model (Actor, Use Case, Use-Case Package) ◆ Use-Case Modeling Guidelines [provided] ◆ Use-Case Priority List ◆ Vision [provided]
7—Analysis	Define an Initial Architecture Analyze Behavior	<ul style="list-style-type: none"> ◆ Architectural Analysis ◆ Review the Architecture ◆ Model the User Experience ◆ Use-Case Analysis ◆ Review the Analysis 	<ul style="list-style-type: none"> ◆ Change Request ◆ Data Model ◆ Deployment Model ◆ Design Guidelines⁵ [provided]
8—Design	Refine the Architecture Detail the Design	<ul style="list-style-type: none"> ◆ Identify Design Mechanisms ◆ Identify Design Elements ◆ Incorporate Existing Design Elements ◆ Describe Concurrency and Distribution ◆ Review the Architecture ◆ Use-Case Design ◆ Subsystem Design ◆ Component Design ◆ Class Design ◆ Database Design ◆ Review the Design 	<ul style="list-style-type: none"> ◆ Design Model (Analysis Class, Design Class, Framework Component, Design Package, Design Subsystem, Interface, Use-Case Realization) ◆ Reference Architecture [provided] ◆ Review Record ◆ Software Architecture Document ◆ User-Experience Guidelines [provided] ◆ User-Experience Model (Navigation Map, Screen, Use-Case Storyboard)
9—Implementation	Structure the Implementation Model Implement Design Elements	<ul style="list-style-type: none"> ◆ Structure the Implementation Model ◆ Implement Design Elements ◆ Perform Unit Tests ◆ Review the Implementation 	<ul style="list-style-type: none"> ◆ Change Request ◆ Implementation Guidelines [provided] ◆ Implementation Model (Implementation File, Implementation Directory) ◆ Programming Guidelines [provided] ◆ Review Record ◆ Test Guidelines [provided] ◆ Test Results ◆ Test Script

⁴ **Change Requests** can be produced during any of the J2EE Developer Roadmap activities. However, they are only explicitly listed as output artifacts from the review activities. The processing of **Change Requests** is not in the scope of the roadmap.

⁵ The provided **Design Guidelines** are refined to include design mechanism details.



are just not the primary focus of the J2EE developer. We will briefly discuss some of these elements in Chapter 10, Additional Topics. For additional information on the process elements that are not included in the J2EE Developer Roadmap, see RUP.

In addition to the detailed process element descriptions provided in the process chapters (Chapters 6-9), we have provided additional information in the appendices that summarizes some key aspects of the elements in the J2EE Developer Roadmap. Specifically, we have provided the following appendices:

- Appendix A, Describing a Software Architecture, which describes what should be considered when documenting the software architecture
- Appendix B, Modeling Conventions, which summarizes the modeling conventions we use throughout the book, including modeling element stereotypes and recommended model structures
- Appendix C, Glossary, which provides brief descriptions of all J2EE Developer Roadmap elements, including artifacts and roles

RUP is an iterative process. Thus, the J2EE Developer Roadmap activities should be interpreted as occurring in the context of phases and iterations. This means that the described activities may be performed multiple times with different emphases and varying levels of effort throughout the software development process. The iterative nature of the disciplines is discussed in more detail in the individual process chapters (Chapters 6-9). For more information on phases and iterations, see Chapter 3.




J2EE Developer Roadmap—J2EE-Specific Content

In this section, we provide a glimpse into the J2EE-specific content provided in the J2EE Developer Roadmap. Table 4.3 summarizes this content by chapter, workflow detail, and activity. In addition to demonstrating the breadth of the J2EE coverage provided in the roadmap, this table can be used to access those parts of the book that address a specific J2EE process area.

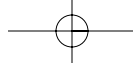
Throughout this book, discipline and workflow detail diagrams are used as the visual representation of the J2EE Developer Roadmap. Examples of these diagrams are provided in Chapter 3. These diagrams serve several important purposes.

- They provide a graphical overview of the process described in the text.

Table 4.3 J2EE-Specific Content of the J2EE Developer Roadmap

 Chapter	 Workflow Detail	 Activity ⁶	J2EE-Specific Content
6—Requirements	Define the System	Capture a Common Vocabulary	None
		Find Actors and Use Cases Prioritize Use Cases	None None
	Refine the System Definition	Detail a Use Case Structure the Use-Case Model	None None
7—Analysis	Define an Initial Architecture	Architectural Analysis	Select a J2EE deployment configuration. Identify possible J2EE technologies that may be used.
	Analyze Behavior	Model the User Experience Use-Case Analysis	None None
8—Design	Refine the Architecture	Identify Design Mechanisms	Identify what J2EE patterns are going to be used.
		Identify Design Elements	Identify what J2EE technologies are going to be used.
	Incorporate Existing Design Elements	Identify JSPs, servlets, EJBs, and other J2EE elements.	
	Detail the Design	Incorporate Existing Design Elements	None
Describe Concurrency and Distribution		Describe the use of Java threads and message-driven EJBs. Map J2EE modules to nodes.	
	Detail the Design	Use-Case Design	Describe the interactions between collaborating J2EE elements.
		Subsystem Design	Describe subsystems in terms of their internal J2EE elements.
		Component Design	Produce a detailed design of EJBs.
		Class Design	Produce a detailed design of JSPs, servlets, and other Java classes.
	Database Design	Database Design	Define the mapping between entity EJBs and the underlying database.
9—Implementation	Structure the Implementation Model	Structure the Implementation Model	Decide on the organization of virtual directory elements (such as JSPs) and Java elements (such as servlets, EJBs, and Java classes), and also the content of J2EE modules.
	Implement Design Elements	Implement Design Elements Perform Unit Tests	Implement EJBs, JSPs, servlets and other Java classes. Create the J2EE modules and their associated deployment descriptors. Test the J2EE elements.

⁶ Review activities are not shown since all review activities are the same, irrespective of the technology used.



- They establish the context of the process chapters, providing “you are here” reference points.
- They describe the overall flow between activities and highlight the key artifacts that are consumed and produced by the activities.

Summary

In this chapter we provided an introduction to the J2EE Developer Roadmap, which is the process we use in this book to guide you through the development of a J2EE application. The process details are provided in the process chapters (Chapters 6-9). The roadmap serves as the organizational guide through the content of these chapters.

Before leaving this chapter, let us emphasize that the J2EE Development Roadmap is a role-based view of RUP that focuses on providing guidance to the J2EE developer. Since its intent is not to cover all aspects of the software development process, we encourage you to refer to RUP for more complete coverage. However, for J2EE developers who want to understand how to apply RUP's best practices in the context of a J2EE software development project, the J2EE Developer Roadmap offers an excellent place to start.

