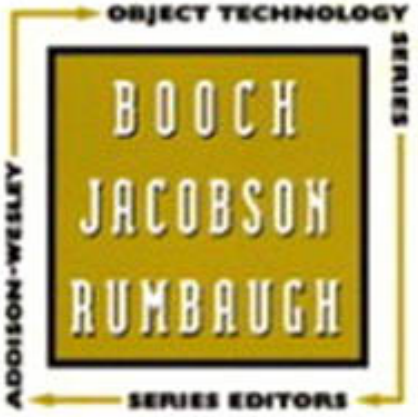
# The Rational Unified Process Made Easy

## A Practitioner's Guide to the RUP

PER KROLL
PHILIPPE KRUCHTEN


Foreword by Grady Booch



# The Rational Unified Process Made Easy

## Detail Key Actors and Use Cases

Another step in understanding what you must build is to refine some of the use cases. At the end of the workshop or brainstorming session, you assign one or several use cases to each analyst, who will describe in further detail essential or critical use cases identified in step 7. Typically, you'll generate a couple of pages for each one. The higher the ceremony, the more detailed the description.

For more information on detailing actors and use cases, see the section Detail Actors and Use Cases, in Chapter 15.

*In parallel, you should also develop user-interface prototypes.*

In parallel to writing the use-case descriptions, you should also develop user-interface prototypes. This allows you to visualize the flow of events, making sure that you, the users, and other stakeholders understand and agree on the flow of events. See the section Develop User-Interface Prototypes, in Chapter 15, for more information.

You should time-box the activities on use cases to avoid getting bogged down into too much detail. You should also focus on capturing the most essential flow of events and point out the existence of alternative flows of events (because it will help you to assess how complex the use case is to implement), rather than describing the details of each flow of events.

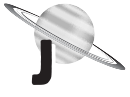
Especially for small projects, you often have the same person take on the role of analyst and developer, meaning that the person who describes a use case will also implement it. If so, you may want to spend less time on documenting the detailed requirements and come back to them as you implement the use case. It is still very useful to identify alternative flows of events, since this will be useful to you when you estimate the amount of work remaining.

For each of our three example projects, the teams do the following:


- Project Ganymede, a small green-field project: The project manager/architect spends a day writing a three-page Vision Document. The team spends half a day in a brainstorming session to come up with an initial set of actors and use cases. Thirteen use cases are found, and each team member takes four to five use cases, spending 30 minutes to detail each use case. Then they get

together and realize that they need another four use cases. They merge two use cases and eliminate one. They spend another two hours describing each of the four most critical use cases (see the section Objective 2: Identify Key System Functionality).


- Project Mars, a large green-field project: In the first iteration, the analysts, involving key stakeholders as necessary, spend roughly a week writing a first-draft Vision of eight pages. A lot of time is spent getting buy-in from all stakeholders. The team spends two days on a use-case workshop with eight key stakeholders, allowing them to come up with a good first-draft use-case model and glossary. In the second iteration, the team refines the Vision, which will be further refined later in the project, especially in Elaboration. They spend roughly four hours describing each of the nine most critical use cases (see the following section, Objective 2: Identify Key System Functionality). In conjunction with doing that, they make a number of updates to the use-case model.




- Project Jupiter, a second generation of a large project: The team makes some updates to the existing Vision, clearly labeling what will be accomplished in the second generation, which is done in a day or two. Most of the time is spent on making an inventory of additional capabilities to implement and the known problems in the first system that must be fixed. The team tries to identify use cases that need to be added, and the most critical of the new use cases are detailed, with the team spending two to four hours on each of them. Planned improvements to existing use cases are listed, but these improvements are normally not detailed at this stage.

## Objective 2: Identify Key System Functionality

*It is important to spend more time up front on the most critical use cases.*

This is the second objective in the Inception phase, and you should work on it as you identify your use cases. It is important to decide which use cases are the most essential or architecturally significant to ensure that you spend more time up front on the most critical use cases.

The project manager and architect should work hand-in-hand on this activity, involving other stakeholders (such as the customer) as necessary, and using several criteria to determine which use cases are critical.

A. **The functionality is the core of the application, or it exercises key interfaces of the system,** and will hence have a major impact on the architecture. Typically an architect identifies these use cases by analyzing redundancy management strategies, resource contention risks, performance risks, data security strategies, and so on. For example, in a Point-Of-Sale system, Check Out and Pay would be a key use case because it validates the interface to a credit-card validation system—it is also critical from a performance and load perspective.
B. **The functionality *must* be delivered.** The functionality captures the essence of the system, and delivering the application without it would be fruitless. Typically the domain and subject-matter experts know what this functionality is from the user perspective (primary behaviors, peak data transaction, critical control transactions, and so on). For example, you cannot deliver an order-entry system if you cannot enter an order.
C. **The functionality covers an area of the architecture that is not covered by any other critical use case.** To ensure that you address all major technical risks, you need to have a good enough understanding of each area of the system. Even if a certain area of the architecture does not seem to be of high risk, it may conceal unexpected technical difficulties that can be exposed by designing, implementing, and testing some of the functionality within that part.

Items A and C will be of greater concern to the architect; project managers will focus mainly on items A and B.

*For a system with 20 use cases, typically only 3 to 4 of them are really critical.*

For a system with 20 use cases, typically 3 to 4 of them are critical.[1] During Inception, it is important to understand *all* the critical use cases you identify and provide a fairly detailed description of them. However, you may postpone describing some of the alternative flows for these critical use cases until later in the project, as long as they do not have a major impact on the architecture.

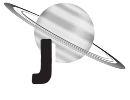
1. Note that for some systems, one or two use cases may constitute the core of the application, with a larger number of "supporting" use cases enabling execution of the core use cases. In this situation, fewer than 20 to 30 percent of use cases are architecturally significant, and we would typically implement several scenarios for each core use case.

The critical use cases are listed in the Software Architecture Document (SAD; see Chapter 16).

For each of our three example projects, you do the following:


- Project Ganymede, a small green-field project: The architect and the project manager is the same person. The architect/project manager suggests 4 of the 15 identified use cases as being critical. After discussion with the customer, a fifth use case is added. The architect/project manager gets the entire team together and spends an hour explaining why these are the most critical use cases. The team agrees, with potential changes made, and the architect/project manager documents the critical use cases in the SAD.


- Project Mars, a large green-field project: The architect proposes a list of 8 of the 40 use cases as being architecturally significant, strictly from a technical risk mitigation standpoint. The project manager suggests a set of 9 use cases that are critical to the stakeholders. The project manager would like stakeholder buy-in on the functionality of these as soon as possible. Five of the use cases overlap. After a few days of discussion between the architect and project manager, the project manager drops 2 use cases from the list (the project can delay getting feedback on those use cases from the users). The architect sees a way to mitigate some of the risks in 8 of the use cases through some of the use cases the project manager added. They end up with a joint list of 9 use cases, which the architect documents in the SAD.




- Project Jupiter, a second generation of a large project: A lot of time will be spent on improving existing use cases, which means that fewer use cases will be critical than for green-field development. The architect identifies one of the existing use cases as critical because it involves using some unexplored new technology. The architect also identifies 2 of the 9 new use cases as being architecturally significant. The project manager identifies one additional use case as critical from the user perspective. The architect documents the 4 critical use cases in the SAD.

## Objective 3: Determine at Least One Possible Solution

Since the overall goal of Inception is to determine whether it makes sense to continue with the project, you need to make sure that there is at least one potential architecture that will allow you to build the system with a sensible amount of risk and at reasonable cost. As an example, you may consider three options for a client/server architecture (see Figure 6.3). By analyzing desired functionality (in the first version, as well as future versions of the application), compatibility with other applications, and requirements on operations and

*Make sure that there is at least one potential architecture that will allow you to build the system with a sensible amount of risk and at reasonable cost.*

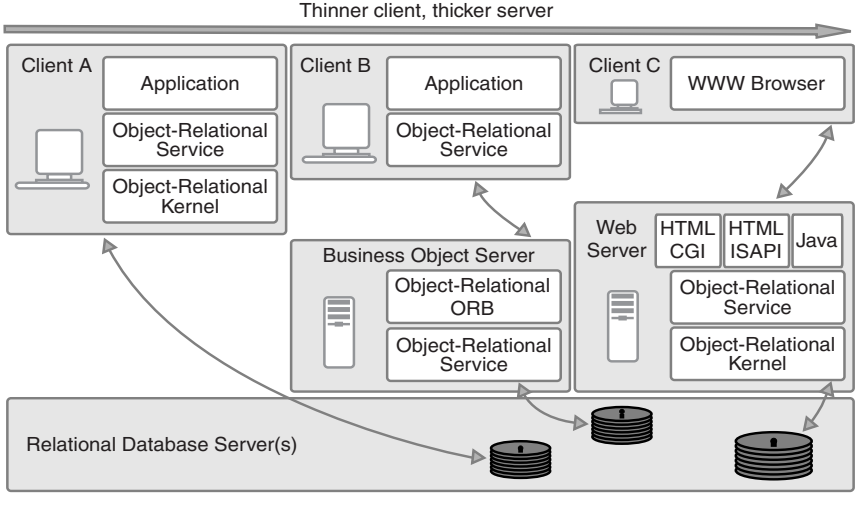



**FIGURE 6.3 Three Options for a Client/Server Architecture.** *During Inception, identify the type of architecture you intend to have and make implementations of necessary elements to the architecture to understand what risks you are facing. Here you see three options for a client/server architecture, each with vastly different requirements for tooling, competency, and complexity, and with different ability to address existing and future requirements on functionality, operation, and maintenance cost.*

maintenance, you may conclude which of these three options are viable. As you explore options, ask the following questions:

- What other, similar systems have been built, and what technology and architecture did you use? What was your cost?
- In particular, for an evolution of an existing system, is the current architecture still satisfactory, or does it need to evolve?
- What technologies would you have to use within the system? Do you need to acquire any new technologies? What are the costs and risks associated with that?
- What software components are needed within the system (database, middleware, and so on)? Can they be purchased? Can they be reused from another in-house project? What are the estimated costs? The associated risks?

In some cases, you may need to acquire or implement some key elements of the architecture, or different suggested architectures, to better understand the risks you are facing and the options you have. For applications where stakeholders might find difficulty envisioning the end product, you should also spend time on implementing some functional prototypes, sufficiently rich to verify that the Vision makes sense.

*At the end of Inception, you should have a rough idea of what risks you are facing.*

At the end of Inception, you should have a rough idea of what risks you are facing, especially in the areas of acquisition of technology and reusable assets, such as architectural framework, packaged software, and so on. During Elaboration, you may come up with a better architecture, and that is fine. It is during Elaboration that you will address the vast majority of the architecture- and technology-related risks you identified during Inception.

For each of our three example projects, you do the following:

- Project Ganymede, a small green-field project: The team builds a functional prototype of the use case that is considered the most critical. The functional prototype identifies some key architectural components, including one that you need to purchase. The team builds bits and pieces of functionality to understand how some new technology can be used, allowing you to better understand what can be delivered.