ISSN 0975 3303

MJS, Vol.1, No.1, May-Oct. 2002, pp 81 - 94

https://doi.org/10.12725/mjs.1.8 | 81-94

RATIONAL UNIFIED PROCESS PART-I

Best Practices for Software Development Teams

Manoj Tharian *

Abstract

This paper presents an overview of the Rational Unified Process® The Rational Unified Process is a software engineering process, delivered through a web-enabled, searchable knowledge base. The process enhances team productivity and delivers software best practices via guidelines, templates and tool mentors for all critical software lifecycle activities. The knowledge base allows development teams to gain the full benefits of the industry-standard Unified Modeling Language (UML).

The Rational Unified Process® is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget. [11, 13]

The Rational Unified Process is a process product, developed and maintained by Rational® Software. The development team for the Rational Unified Process are working closely with customers, partners, Rational's product groups as well as Rational's consultant organization, to ensure that the process is continuously updated and improved upon to reflect recent experiences and evolving and proven best practices.

The Rational Unified Process is a guide for how to effectively use the Unified Modeling Language (UML). The UML is a industry-standard language that allows us to clearly communicate requirements, architectures and designs. The UML was originally created by Rational Software, and is now maintained by the standards organization Object Management Group (OMG). [4]

* Director, MicroGenesis TechSoft, 20,1st Cross, Vasanthnagar, Bangalore – 560 052, INDIA. email: manoj.tharian@mgenindia.com

The Rational Unified Process captures many of the best practices in modern software development in a form that is suitable for a wide range of projects and organizations. Deploying these best practices¾using the Rational Unified Process as your guide¾offers development teams a number of key advantages. In next section, we describe the six fundamental best practices of the Rational Unified Process.

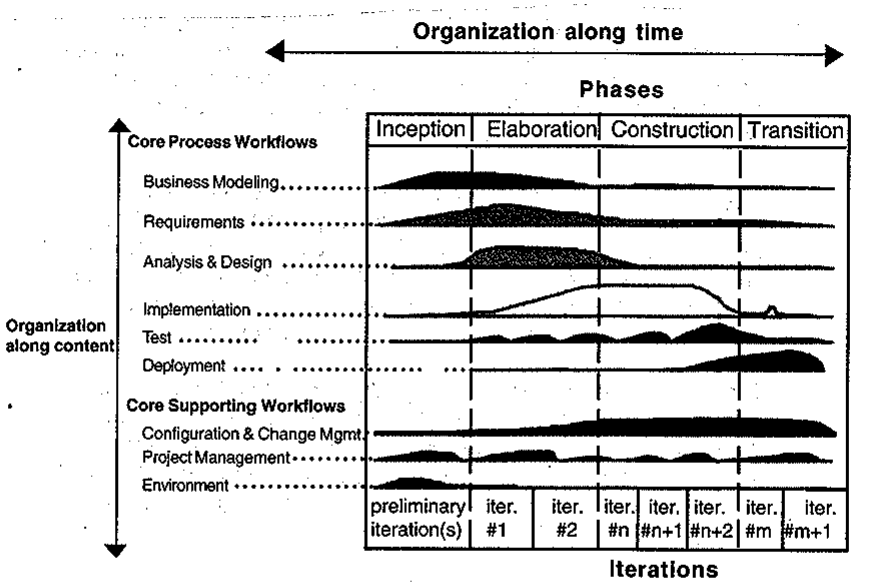
The Rational Unified Process describes how to effectively deploy commercially proven approaches to software development for software development teams. These are called "best practices" not so much because you can precisely quantify their value, but rather, because they are observed to be commonly used in industry by successful organizations.

Introduction

Two Dimensions

The process can be described in two dimensions, or along two axis:

- the horizontal axis represents time and shows the dynamic aspect of the process as it is enacted, and it is expressed in terms of cycles, phases, iterations, and milestones.
- the vertical axis represents the static aspect of the process: how it is described in terms of activities, artifacts, workers and workflows.



The Iterative Model graph shows how the process is structured along two dimensions.

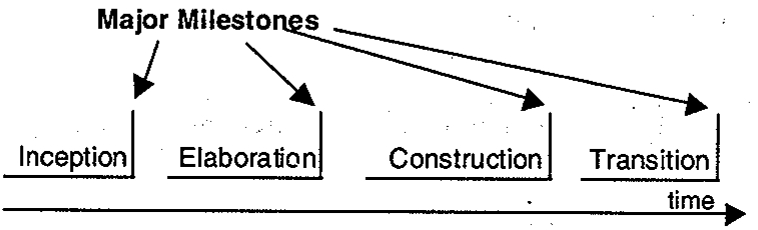
Phases and Iterations - The Time Dimension

This is the dynamic organization of the process along time.

The software lifecycle is broken into cycles, each cycle working on a new generation of the product. The Rational Unified Process divides one development cycle in four consecutive phases [10]

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Each phase is concluded with a well-defined milestone—a point in time at which certain critical decisions must be made, and therefore key goals must have been achieved [2].



The phases and major milestones in the process.

Each phase has a specific purpose.

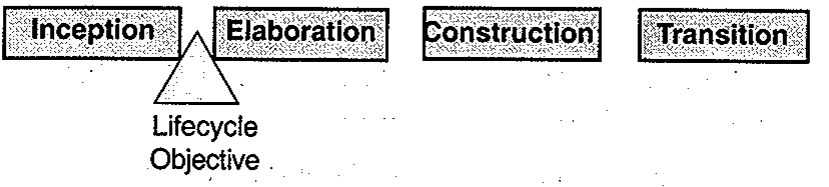
Inception Phase

During the inception phase, you establish the business case for the system and delimit the project scope. To accomplish this you must identify all external entities with which the system will interact (actors) and define the nature of this interaction at a high-level. This involves identifying all use cases and describing a few significant ones. The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones. [10, 14]

The outcome of the inception phase is:

- A vision document: a general vision of the core project's requirements, key features, and main constraints.
- An initial use-case model (10%-20% complete).
- An initial project glossary (may optionally be partially expressed as a domain model).
- An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast.
- An initial risk assessment.
- A project plan, showing phases and iterations.
- A business model, if necessary.
- One or several prototypes.

Milestone : Lifecycle Objectives



At the end of the inception phase is the first major project milestone: the Lifecycle Objectives Milestone. The evaluation criteria for the inception phase are:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Actual expenditures versus planned expenditures.

The project may be cancelled or considerably re-thought if it fails to pass this milestone.

Elaboration Phase

The purpose of the elaboration phase is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project. To accomplish these objectives, you must have the "mile wide and inch deep" view of the system. Architectural decisions have to be made with an understanding of the whole system: its scope, major functionality and nonfunctional requirements such as performance requirements.

It is easy to argue that the elaboration phase is the most critical of the four phases. At the end of this phase, the hard "engineering" is considered complete and the project undergoes its most important day of reckoning: the decision on whether or not to commit to the construction and transition phases. For most projects, this also corresponds to the transition from a mobile, light and nimble, low-risk operation to a high-cost, high-risk operation with substantial inertia. While the process must always accommodate changes, the elaboration phase activities ensure that the architecture, requirements and plans are stable enough, and the risks are sufficiently mitigated, so you can predictably determine the cost and schedule for the completion of the development. Conceptually, this level of fidelity would correspond to the level necessary for an organization to commit to a fixed-price construction phase.

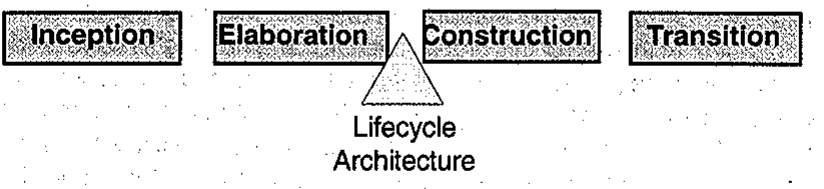
In the elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk, and novelty of the project. This effort should at least address the critical use cases identified in the inception phase, which typically expose the major technical risks of the project. While an evolutionary prototype of a production-quality component is always the goal, this does not exclude the development of one or more exploratory, throw-away prototypes to mitigate specific risks such as design/requirements trade-offs, component feasibility study, or demonstrations to investors, customers, and end-users.

The outcome of the elaboration phase is:

- A use-case model (at least 80% complete) — all use cases and actors have been identified, and most use-case descriptions have been developed.
- Supplementary requirements capturing the non functional requirements and any requirements that are not associated with a specific use case.

- A Software Architecture Description.
- An executable architectural prototype.
- A revised risk list and a revised business case.
- A development plan for the overall project, including the coarse-grained project plan, showing iterations" and evaluation criteria for each iteration.
- An updated development case specifying the process to be used.
- A preliminary user manual (optional).

Milestone : Lifecycle Architecture



At the end of the elaboration phase is the second important project milestone, the Lifecycle Architecture Milestone. At this point, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks.

The main evaluation criteria for the elaboration phase involves the answers to these questions:

- Is the vision of the product stable?
- Is the architecture stable?
- Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
- Is the plan for the construction phase sufficiently detailed and accurate? Is it backed up with a credible basis of estimates?
- Do all stakeholders agree that the current vision can be achieved if the current plan is executed to develop the complete system, in the context of the current architecture?
- Is the actual resource expenditure versus planned expenditure acceptable?

The project may be aborted or considerably re-thought if it fails to pass this milestone.

Construction Phase

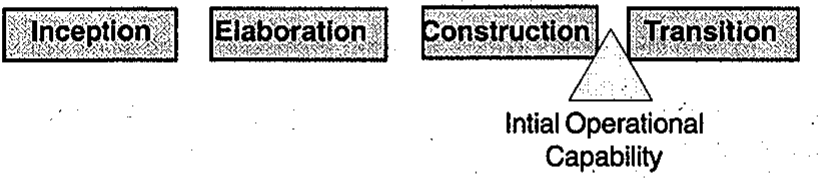
During the construction phase, all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. The construction phase is, in one sense, a manufacturing process where emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality. In this sense, the management mindset undergoes a transition from the development of intellectual property during inception and elaboration, to the development of deployable products during construction and transition.

Many projects are large enough that parallel construction increments can be spawned. These parallel activities can significantly accelerate the availability of deployable releases; they can also increase the complexity of resource management and workflow synchronization. A robust architecture and an understandable plan are highly correlated. In other words, one of the critical qualities of the architecture is its ease of construction. This is one reason why the balanced development of the architecture and the plan is stressed during the elaboration phase.

The outcome of the construction phase is a product ready to put in hands of its end-users. At minimum, it consists of:

- The software product integrated on the adequate platforms.
- The user manuals.
- A description of the current release.

Milestone : Initial Operational Capability



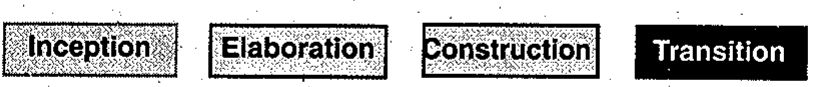
At the end of the construction phase is the third major project milestone (Initial Operational Capability Milestone). At this point, you decide if the software, the sites, and the users are ready to go operational, without exposing the project to high risks. This release is often called a "beta" release.

The evaluation criteria for the construction phase involve answering these questions:

- Is this product release stable and mature enough to be deployed in the user community?
- Are all stakeholders ready for the transition into the user community?
- Are the actual resource expenditures versus planned expenditures still acceptable?

Transition may have to be postponed by one release if the project fails to reach this milestone.

Transition Phase



The purpose of the transition phase is to transition the software product to the user community. Once the product has been given to the end user, issues usually arise that require you to develop new releases, correct some problems, or finish the features that were postponed.

The transition phase is entered when a baseline is mature enough to be deployed in the end-user domain. This typically requires that some usable subset of the system has been completed to an acceptable level of quality and that user documentation is available so that the transition to the user will provide positive results for all parties. This includes:

- "beta testing" to validate the new system against user expectations
- parallel operation with a legacy system that it is replacing
- conversion of operational databases
- training of users and maintainers
- roll-out the product to the marketing, distribution, and sales teams

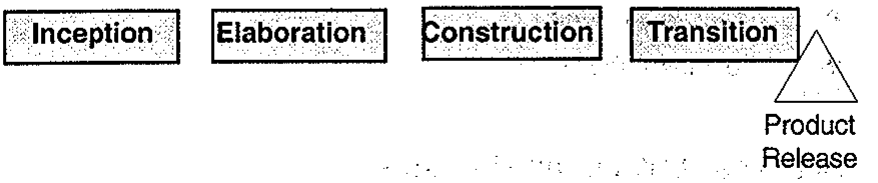
The transition phase focuses on the activities required to place the software into the hands of the users. Typically, this phase includes several iterations, including beta releases, general availability releases, as well as bug-fix and enhancement releases. Considerable effort is expended in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback. At this point in the lifecycle, however, user feedback should be confined primarily to product tuning, configuring, installation, and usability issues.

The primary objectives of the transition phase include:

- Achieving user self-supportability
- Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision.
- Achieving final product baseline as rapidly and cost effectively as practical

This phase can range from being very simple to extremely complex, depending on the type of product. For example, a new release of an existing desktop product may be very simple, whereas replacing a nation's air-traffic control system would be very complex.

Milestone: Product Release



At the end of the transition phase is the fourth important project milestone, the Product Release Milestone. At this point, you decide if the objectives were met, and if you should start another development cycle. In some cases, this milestone may coincide with the end of the inception phase for the next cycle.

The primary evaluation criteria for the transition phase involve the answers to these questions:

- Is the user satisfied?
- Are the actual resources expenditures versus planned expenditures still acceptable?

Iterations

Each phase in the Rational Unified Process can be further broken down into iterations. An iteration is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows incrementally from iteration to iteration to become the final system [10].

Benefits of an iterative approach

Compared to the traditional waterfall process, the iterative process has the following advantages:

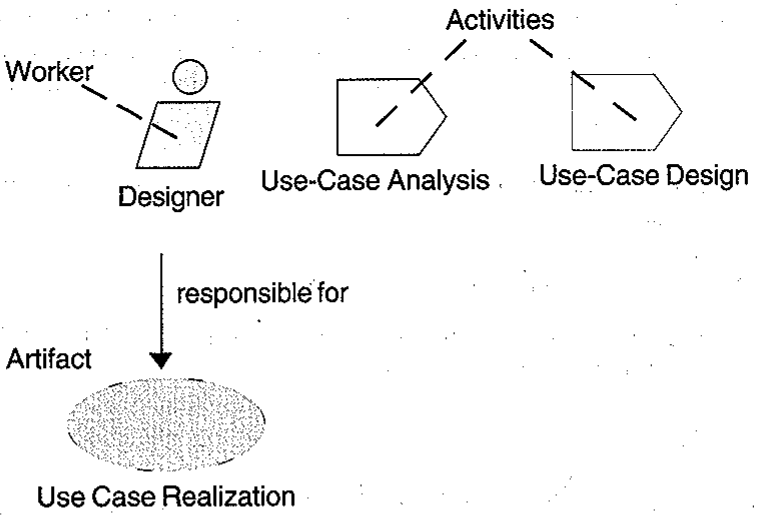
- Risks are mitigated earlier
- Change is more manageable
- Higher level of reuse
- The project team can learn along the way
- Better overall quality

Static Structure of the Process

A process describes who is doing what, how, and when. The Rational Unified Process is represented using four primary modeling elements:

- Workers, the 'who'
- Activities, the 'how'
- Artifacts, the 'what'
- Workflows, the 'when'

Activities, Artifacts, and Workers

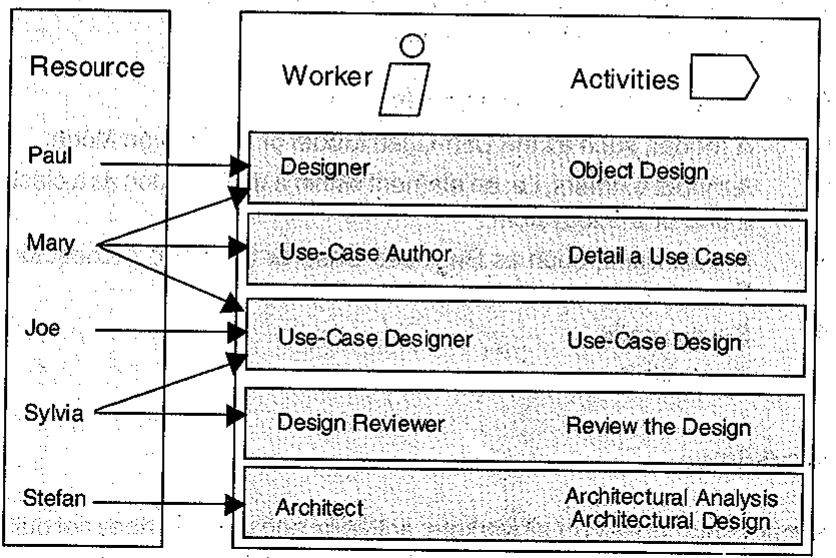


Workers, activities and artifacts.

Worker

A worker defines the behavior and responsibilities of an individual, or a group of individuals working together as a team. You could regard a worker as a "hat" an individual can wear in the project. One individual

may wear many different hats. This is an important distinction because it is natural to think of a worker as the individual or team itself, but in the Unified Process the worker is more the role defining how the individuals should carry out the work. The responsibilities we assign to a worker includes both to perform a certain set of activities as well as being owner of a set of artifacts.



People and Workers

Activity

An activity of a specific worker is a unit of work that an individual in that role may be asked to perform. The activity has a clear purpose, usually expressed in terms of creating or updating some artifacts, such as a model, a class, a plan. Every activity is assigned to a specific worker. The granularity of an activity is generally a few hours to a few days, it usually involves one worker, and affects one or only a small number of artifacts. An activity should be usable as an elements of planning and progress; if it is too small, it will be neglected, and if it is too large, progress would have to be expressed in terms of an activity's parts.

Example of activities:

- Plan an iteration, for the Worker : Project Manager
- Find use cases and actors, for the Worker : System Analyst
- Review the design, for the Worker : Design Reviewer
- Execute performance test, for the Worker : Performance Tester

Artifact

An artifact is a piece of information that is produced, modified, or used by a process. Artifcats are the tangible products of the project, the things the project produces or uses while working towards the final product. Artifacts are used as input by workers to perform an activity, and are the result or output of such activities. In object-oriented design terms, as activities are operations on an active object (the worker), artifacts are the parameters of these activities.

Artificats may take various shapes or forms:

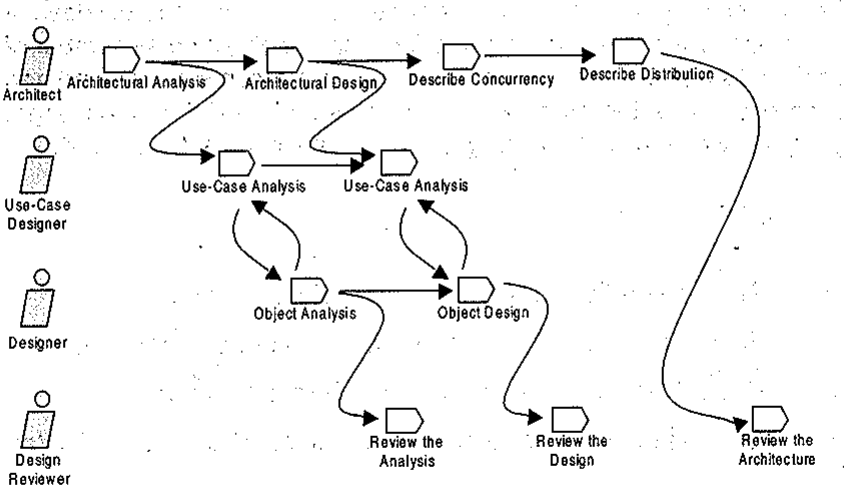
- A model, such as the Use-Case Model or the Design Model
- A model element, i.e. an element within a model, such as a class, a use or a subsystem
- A document, such as Business Case or Software Architecture Document
- Source code
- Executables

Workflows

A mere enumeration of all workers, activities and artifacts does not quite constitute a process. We need a way to describe meaningful sequences of activities that produce some valuable result, and to show interactions between workers.

A worklfow is a sequence of activities that produces a result of observable vlaue.

In UML terms, a workflow can be expressed as a sequence diagram, a collaboration diagram, or an activity diagram. We use a form of activity diagrams in this white paper.



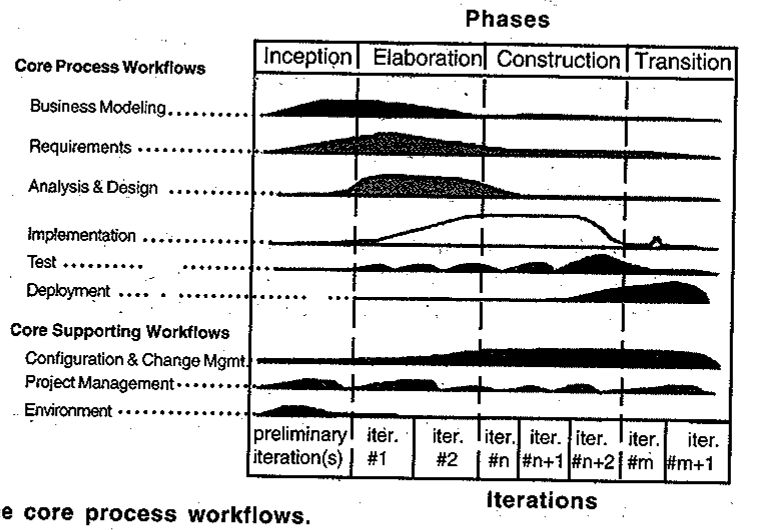
Example of workflow.

Note that it is not always possible or practical to represent all of the dependencies between activities. Often two activities are more tightly interwoven than shown, especially when they involve the same worker or the same individual. People are not machines, and the workflow cannot be interpreted literally as a program for people, to be followed exactly and mechanically.

In next section we will discuss the most essential type of workflows in the process, called Core Workflows.

Core workflows

There are nine core process workflows in the Rational Unified Process, which represent a partitioning of all workers and activities into logical groupings.



The nine core process workflows.

The core process workflows are divided into six core "engineering" workflows:

1. Business modeling workflow
2. Requirements workflow
3. Analysis & Design workflow
4. Implementation workflow
5. Test workflow
6. Deployment workflow

And three core "supporting" workflows:

1. Project Management workflow
2. Configuration and Change Management workflow
3. Environment workflow

Although the names of the six core engineering workflows may evoke the sequential phases in a traditional waterfall process, we should keep in mind that the phases of an iterative process are different and that these workflows are revisited again and again throughout the lifecycle. The actual complete workflow of a project interleaves these nine core workflows, and repeats them with various emphasis and intensity at each iteration.

.....to be continued in the next issue

References

1. Barry W. Boehm, A Spiral Model of Software Development and Enhancement, Computer, May 1988, IEEE, pp.61-72
2. Barry W. Boehm, Anchoring the Software Process, IEEE Software, 13, 4, July 1996, pp. 73-82.
3. Grady Booch, Object Solutions, Addison-Wesley, 1995.
4. Grady Booch, Ivar Jacobson, and James Rumbaugh, Unified Modeling Language 1.3, White paper, Rational Software Corp., 1998.
5. Alan W. Brown (ed.), Component-Based Software Engineering, IEEE Computer Society, Los Alamitos, CA, 1996, pp.140.
6. Michael T. Devlin, and Walker E. Royce, Improving Software Economics in the Aerospace and Defense Industry, Technical paper TP-46, Santa Clara, CA, Rational Software Corp., 1995
7. Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard, Object-Oriented Software Engineering—A Use Case Driven Approach, Wokingham, England, Addison-Wesley, 1992, 582p.
8. Ivar Jacobson, M. Griss, and P. Jonsson, Software Reuse—Architecture, Process and Organization for Business Success, Harlow, England, AWL, 1997.
9. Philippe Kruchten, The 4+1 View Model of Architecture, IEEE Software, 12 (6), November 1995, IEEE, pp.42-50.
10. Philippe Kruchten, A Rational Development Process, CrossTalk, 9 (7), STSC, Hill AFB, UT, pp.11-16.
11. Ivar Jacobson, Grady Booch, and Jim Rumbaugh, Unified Software Development Process, Addison-Wesley, 1999.
12. Grady Booch, Jim Rumbaugh, and Ivar Jacobson, Unified Modeling Language—User's Guide, Addison-Wesley, 1999.
13. Philippe Kruchten, Rational Unified Process—An Introduction, Addison-Wesley, 1999.
14. Walker Royce, Software Project Management—A Unified Framework, Addison-Wesley, 1998.