Whitepaper
Collection

Volume

5

Rational Unified Process®

An Overview

Published by Senior Consulting
Strategical, Conceptual and Technical Consulting

Alexander Gola
Gröbenzeller Strasse 37
82178 Puchheim, Germany
phone +49 89 84050934
mobile +49 172 5473831
e-mail alex@alexander-gola.de
web www.alexander-gola.de

Abstract

This paper describes the Rational Unified Process® (RUP®), which is a software design methodology created by the Rational Software Corporation. 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.

Modification History

Version	Date	Author	Description of Changes
0.1	12.02.2004	Alexander Gola	Initial Release
1.0	20.02.2004	Alexander Gola	First Release
1.1	21.02.2004	Alexander Gola	Layout changes
1.2	29.05.2004	Alexander Gola	Changes
1.3	12.01.2005	Alexander Gola	Enhancements
1.4	18.01.2005	Alexander Gola	Enhancements, Chapter "Trademarks"
1.5	27.07.2005	Alexander Gola	Changes
1.6	18.10.2005	Alexander Gola	Publishing informations
1.7	26.01.2007	Alexander Gola	Changed company address

Table Of Contents

List Of Figures

List Of Tables

1 The Ten Essentials of a Project

The following list describes the minimal set of items a project will have in place if they are truly following the idea of quality.

1.1 Vision

The vision provides a high-level, sometimes contractual basis for more detailed technical requirements.
The vision captures very high-level requirements and design constraints to give the reader an understanding of the system to be developed. It provides input to the project-approval process and is therefore intimately related to the business case. It communicates the fundamental "why is and what is" related to the project and is a gauge against which all future decisions should be validated.
The contents of the vision should answer the following questions, which might be broken out to separate, more detailed, artifacts, as needed:

- What are the key terms ? (Glossary)
- What problem are we trying to solve ? (Problem Statement)
- Who are the stakeholders ? Who are the users ? What are their needs ?
- What are the product features ?
- What are the functional requirements ? (Use Cases)
- What are the non-functional requirements ?
- What are the design constraints ?

1.2 Plan

A Software Development Plan gathers all information required to manage the project. It may enclose a number of separate artifacts developed during this phase and is maintained throughout the project.
The plan is used to plan the project schedule and resource needs and to track progress against the schedule. It addresses such areas as: Project Organization, Schedule (Project Plan, Iteration Plan, Resources and Tools), Requirements Management Plan, Configuration Management Plan, Problem Resolution Plan, QA Plan, Test Plan, Test Cases, Evaluation Plan and Product Acceptance Plan.
In a simple project, these may include only one or two sentences each.
The format of the plan itself is not as important as the activity and thought that go into producing it.

1.3 Risks

An essential precept of the process is to identify and attack the highest risk items early in the project.
The risk list is intended to capture the perceived risks to the success of the project. It identifies, in decreasing order of priority, the events which could lead to a significant negative outcome.
Along with each risk, should be a plan for mitigating that risk. This serves as a focal point for planning project activities and is the basis around which iterations are organized.

1.4 Issues

Continuous open communication with objective data derived directly from ongoing activities and the evolving product configurations are important in any project.
In a quality process, this is done through regular status assessments, which provide the mechanism for addressing, communicating and resolving management issues, technical issues and project risks. In addition to identifying the issues, each should be assigned a due date, with a responsible person who is accountable for the resolution. This should be regularly tracked and updated as necessary. These project snapshots provide the heartbeat for management attention. While the period may vary, the forcing function needs to capture the project history and resolve to remove any roadblocks or bottlenecks that restrict progress.

1.5 Business Case

The Business Case provides the necessary information, from a business standpoint, to determine whether or not this project is worth investing in.
The main purpose is to develop an economic plan for realizing the project Vision. Once developed, the Business Case is used to make an accurate assessment of the Return On Investment (ROI) provided by the project. It provides the justification for the project and establishes its economic constraints. It provides information to the economic decision makers on the projects worth and is used to determine whether the project should move ahead.
The description should not delve deeply into the specifics of the problem, but rather it should create a compelling argument why the product is needed. It must be brief, however, so that it is easy enough for all project team members to understand and remember. At critical milestones, the Business Case is re-examined to see if estimates of expected return and cost are still accurate and whether the project should be continued.

1.6 Architecture

The architecture of a software system is the organization or structure of the systems significant components interacting through interfaces, with components composed of successively smaller components and interfaces.
A good process provides a methodical, systematic way to design, develop and validate a software architecture. This is the "essence" of an Analysis and Design workflow: defining a candidate architecture, refining the architecture, analyzing behaviour and designing components of the system.
To speak and reason about software architecture, first it must define an architectural representation, a way of describing important aspects of an architecture. This description is captured in the Software Architecture Document, which presents the architecture in multiple views. Each architectural view addresses some specific set of concerns, specific to stakeholders in the development process: end users, designers, managers, system engineers, maintainers and so on. This serves as a communication medium between the architect and other project team members regarding architecturally significant decisions which have been made on the project.

1.7 Product

The "essence" of the Implementation and Test workflows is to incrementally code, build and test the components of the system, with executable releases at the end of each iteration after inception.
At the end of this phase, an architectural prototype is available for evaluation. This might also include a user-interface prototype, if necessary. Throughout each iteration of the next phase, components are integrated into executable, tested builds that evolve into the final product.
Key to this essential element is an integrated set of test activities that accompany the building of the product - as well as ongoing Configuration Management and review activities.

1.8 Evaluation

The Iteration Assessment captures the results of an iteration, the degree to which the evaluation criteria were met, the lessons learned and process changes to be implemented.
The Iteration Assessment is an essential artifact of the iterative approach. Depending on the scope and risk of the project and the nature of the iteration, it may range from a simple record of demonstration and outcomes to a complete formal test review record.

1.9 Change Requests

The "essence" of the Configuration and Change Management workflow is to manage and control the scope of the project, as changes occur throughout the project lifecycle, while maintaining the goal of considering all stakeholder needs and meeting those, to whatever extent possible.
As soon as the first prototype is put before the users (and often even before that), changes will be requested. In order to control those changes and effectively manage the scope of the project and expectations of the stakeholders, it is important that all changes to any development artifacts be proposed through Change Requests and managed with a consistent process.
Change Requests are used to document and track defects, enhancement requests and any other type of request for a change to the product. The benefit of Change Requests is that they provide a record of decisions, and, due to their assessment process, ensure that impacts of the potential change are understood by all project team members. The Change Requests are essential for managing the scope of the project, as well as assessing the impact of proposed changes.

1.10 User Support

The "essence" of the Deployment workflow is to wrap up and deliver the product, along with whatever materials are necessary to assist the end-user in learning, using, operating and maintaining the product.
At a minimum, this should include a User Guide, perhaps implemented through online help and may also include an Installation Guide and Release Notes. Depending on the complexity of the product, training materials may also be needed, as well as a bill of materials along with any product packaging.

2 A Brief History of the Rational Unified Process®

The Rational Unified Process®, or RUP®, has matured over many years and reflects the collective experience of many peoples and companies.

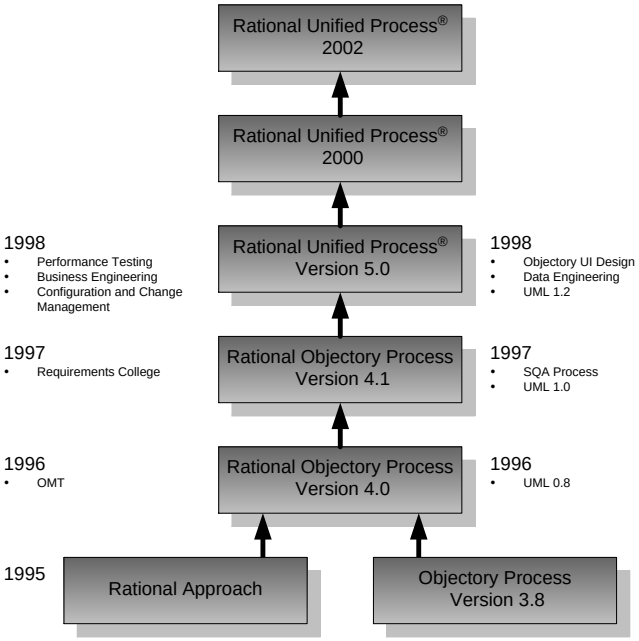

Figure 1 Rational Unified Process® History

Going backwards in time, the Rational Unified Process® is the direct successor to the Rational Objectory Process Version 4. The process incorporates more material in the areas of data engineering, business modelling, project management and configuration management, the latter as a result of the merger with Pure-Atria. It also brings a tighter integration to the Rational Software suite of tools.
The Rational Objectory Process was the result of the integration of the "Rational Approach" and the Objectory Process Version 3, after the merger of Rational Software Corporation and Objectory AB in 1995. From its Objectory ancestry, the process has inherited its process structure and the central concept of use case. From the Rational background, it gained the current formulation of iterative development and architecture. This version also incorporated material on requirements management from Requisite, Inc. and a detailed test process inherited from SQA,® Inc., companies which also merged with Rational Software. Finally, this process was the first one to use the newly created Unified Modeling Language (UML 0.8).
The Objectory Process was created in Sweden in 1987 by Ivar Jacobson as the result of his experience with Ericsson. This process became a product at his company, Objectory AB. Centered around the concept of use case and an object-oriented design method, it rapidly gained recognition in the software industry and has been adopted and integrated by many companies worldwide.

3 Introduction into the Rational Unified Process®

The Rational Unified Process® is a configurable software development process platform that delivers proven best practices and a configurable architecture. The framework provides development teams with a common set of software development best practices. The RUP® can easily be adapted to the specific needs of projects and teams and for each stage of a project.
The RUP® platform includes tools for configuring to project specific needs and for developing own internal knowledge into process.

3.1 Process Framework

It is much more effective to take a more systematic and holistic approach, making sure that the key elements of a process are in place (architecture) before determining to focus on any one particular problem area.
Once this framework (or architecture) for a quality software process is in place, then a project can effectively focus on a particular area which is identified as being a major contributor to their problems.

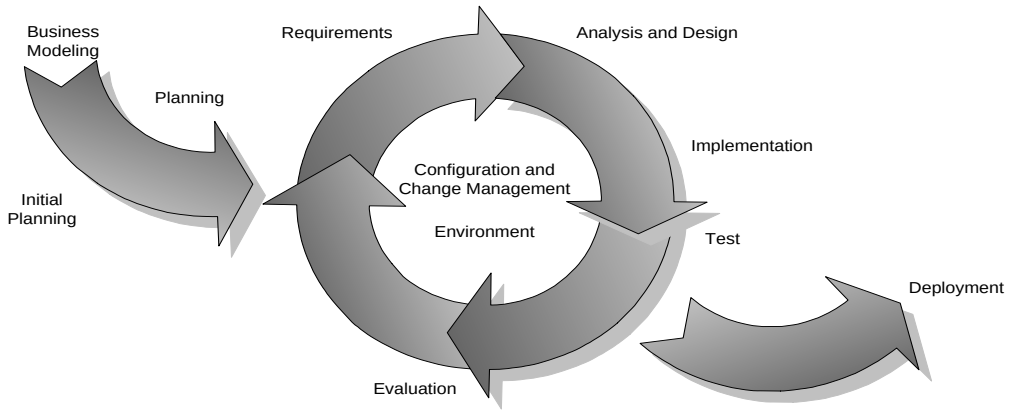

Figure 2 Rational Unified Process® Framework

By adopting the RUP® framework, an organization can shortcut the need to internally develop and document its procedures. By tailoring the RUP® to meet its specific needs, teams can quickly take advantage of proven best practices, templates, guidelines and other assets to create a custom blueprint for project success.
The RUP® helps also to satisfy many goals from the Capability Maturity Model® (CMM). This industry quality model helps to consolidate existing software and systems capability maturity models, helps to reduce complexity and increase the ROI associated with engineering process improvement.
For more informations about the CMM model, see my Whitepaper Collection document called "Capability Maturity Model - An Overview".

3.2 Process Overview

The Rational Unified Process® may be approached from two different and integrated perspectives:

- Phases - a management perspective, dealing with the financial, strategic, commercial and human aspects
- Iterations - a technical perspective, dealing with quality, engineering and design method aspects

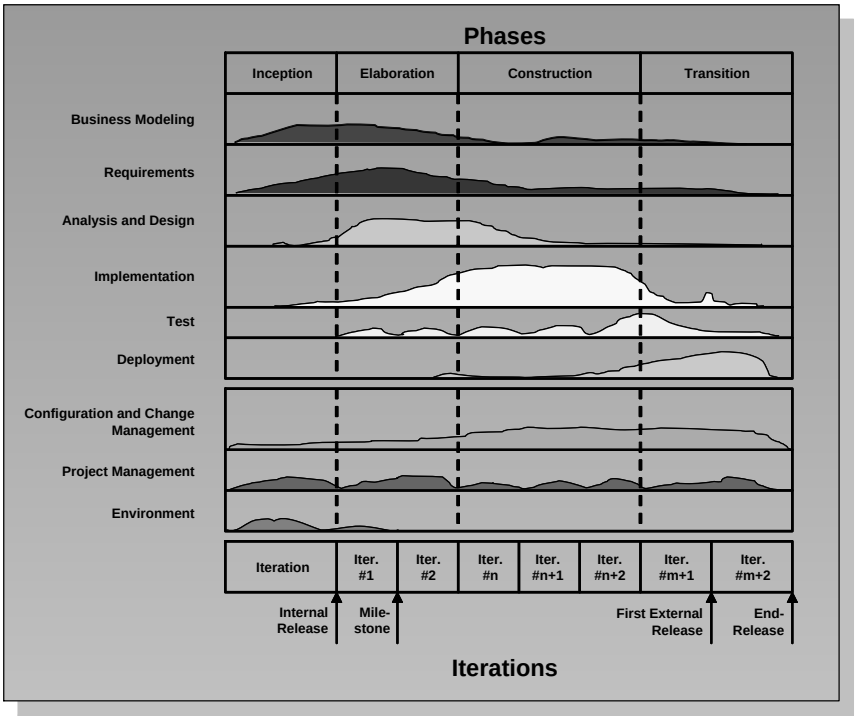

Figure 3 Rational Unified Process® Overview

In an iterative process, the activities of planning, test and integration are spread incrementally throughout the cycle, in each iteration and not massively lumped at the beginning and at the end. They do not appear as separate steps or phases in the process.
Although this will vary considerably depending on the project discriminates, a typical initial development cycle for a medium size project should anticipate the following ratios for various activities:

Activities	Ratio
Planning and Management	15 %
Analysis / Requirements	10 %
Design / Integration	15 %
Implementation / Functional Tests	30 %
Measurement / Assessment / Acceptance Test	15 %
Tools / Environment / Change Management	10 %
Maintenance (fixes during development)	5 %

Table 1 Rational Tools Activities

3.2.1 Management Perspective

As seen from a management perspective, the business and economics point of view, the software life-cycle is organized along four main phases. These phases are the indicators of the project progress:

- Inception - Specifying the end-product vision and its business case, defining the scope of the project.
- Elaboration - Planning the necessary activities and required resources; specifying the features and designing the architecture.
- Construction - Building the product and evolving the vision, the architecture and the plans until the product (the completed vision) is ready for transfer to its user community.
- Transition - Transitioning the product to its user community, which includes manufacturing, delivering, training, supporting, maintaining the product until the users are satisfied.

Going through these phases is called a development cycle and it produces a software generation. Unless the life of the product stops, an existing product will evolve into its next generation by repeating the same sequence of inception, elaboration, construction and transition phases, with a different emphasis however on the various phases. As the product eventually goes through several cycles, new generations are being produced.

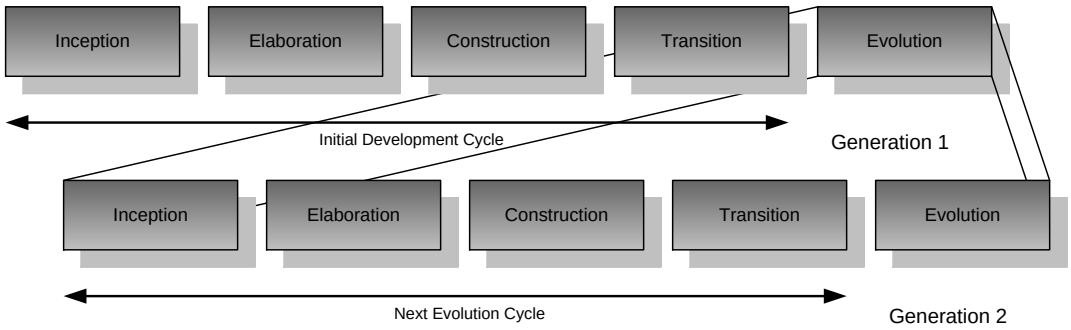

Figure 4 Management Perspective

In practice, cycles may slightly overlap: the inception and elaboration phase may start during the trailing part of the transition phase of the previous cycle.

3.2.2 Technical Perspective

From a technical perspective the software development is seen as a succession of iterations, through which the software under development evolves incrementally.
Each iteration is concluded by the release of an executable product which may be a subset of the complete vision, but useful from some engineering or user perspective. Each release is accompanied by supporting artifacts: release description, user documentation, plans, etc.

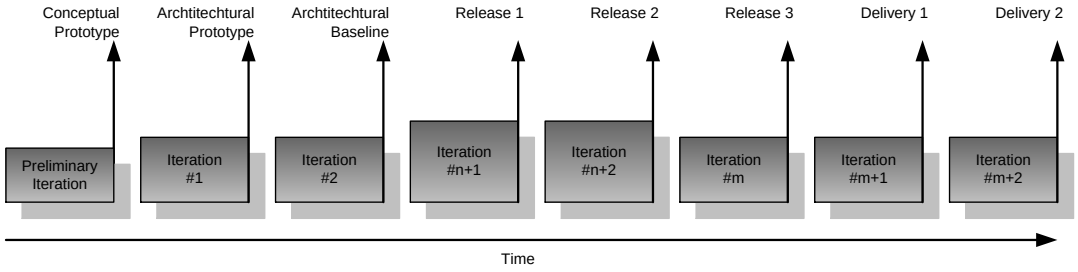

Figure 5 Technical Perspective

A iteration consists of the activities of planning, analysis, design, implementation, testing, in various proportions depending on where the iteration is located in the development cycle.

3.3 Process Phases

3.3.1 Inception Phase

This phase brings to light an original vision of a potential product and transforms it into an actual project. Its purpose is to establish the business case for a new product or a major update and to specify the project scope.
For the development of a new product, the main outcome of this phase is a "go-no go" decision to move into the next phase and to invest time and money to analyze in detail what it to be built, can it be built and how to build it.
For the evolution of an existing product, this may be a simple and short phase, based on users or customers requests, on problem reports and on new technological advances. For a contractual development, the decision to proceed is based on experience of the specific domain and on the competitiveness of the development organization in this domain or market. In this case the inception phase may be concluded by a decision to bid or by the bid itself. The idea may be based on an existing research prototype, whose architecture may or may not be suitable for the final software.

Entry criteria
The expression of a need, which can take any of the following forms:

- an original vision
- a legacy system
- an Request For Proposal (RFP)
- the previous generation and a list of enhancements
- some assets (software, know-how, financial assets)
- a conceptual prototype or a mock-up

Exit criteria

- An initial business case containing at least:
  - a clear formulation of the product vision in terms of functionality, scope, performance, capacity, technology base
  - success criteria (for instance revenue projection)
  - an initial risk assessment
  - an estimate of the resources required to complete the elaboration phase
- Optionally at the end of the inception phase, there are:
  - an initial domain analysis model (~10%-20% complete), identifying the top key use cases and sufficient to drive the architecture effort
  - an initial architectural prototype, which at this stage may be a throw-away prototype

3.3.2 Elaboration Phase

The purpose of this phase is to more thoroughly analyze the problem domain, to define and stabilize the architecture and to address the highest risk elements of the project. So that at the end of the phase the project team can produce a comprehensive plan showing how the two next phases will be done:

- A baseline product vision (i.e., an initial set of requirements) based on an analysis model
- Evaluation criteria for at least the first construction iteration
- A baseline software architecture
- The resources necessary to develop and deploy the product, especially in terms of people and tools
- A schedule
- A resolution of the risks sufficient to make a "high fidelity" cost, schedule and quality estimate of the construction phase.

In this phase an executable architectural prototype is built, in one or several iterations depending on the scope, size, risk and novelty of the project, which addresses at least the top key use cases identified in the inception phase and which addresses the top technical risks of the project. This is an evolutionary prototype, of production quality code which becomes the architectural baseline, but it does not exclude the development of one or more exploratory, throw-away prototypes to mitigate specific risks: refinements of the requirements, feasibility, human-interface studies, demonstrations to investors, etc.
At the end of this phase, there is again a "go-no go" decision point to actually invest and build the product (or bid for the complete development of the contract). The plans produced must be detailed enough and the risks sufficiently mitigated to be able to determine with accuracy the cost and schedule for the completion of the development.

Entry criteria

- The products and artifacts described in the exit criteria of the previous phase
- The plan was approved by the project management, funding authority and the resources required for the elaboration phase have been allocated

Exit criteria

- a detailed software development plan, containing:
  - an updated risk assessment
  - a management plan
  - a staffing plan
  - a phase plan showing the number and contents of the iteration
  - an iteration plan, detailing the next iteration
  - the development environment and other tools required
  - a test plan
- a baseline vision, in the form of a set of evaluation criteria for the final product

- objective, measurable evaluation criteria for assessing the results of the initial iterations(s) of the construction phase
- a domain analysis model (80% complete), sufficient to be able to call the corresponding architecture "complete"
- a software architecture description (stating constraints and limitations)
- an executable architecture baseline

3.3.3 Construction Phase

This phase is broken down into several iterations, fleshing out the architecture baseline and evolving it in steps or increments towards the final product. At each iteration, the various artifacts prepared during the elaboration phase are expanded and revised, but they ultimately stabilize as the system evolves in correctness and completeness.
New artifacts are produced during this phase beside the software itself: documentation, both internal and for the end-users, test beds, test suites and deployment collaterals to support the next phase.

Entry criteria for each iteration

- The product and artifacts of the previous iteration. The plan must state the iteration specific goals:
  - Additional capabilities being developed, e.g. which use cases or scenarios will be covered.
  - Risks being mitigated during this iteration.
  - Defects being fixed during the iteration.

Exit criteria

The same products and artifacts, updated, plus:

- A release description document, which captures the results of an iteration.
- Test cases and results of the tests conducted on the products.
- An iteration plan, detailing the next iteration.
- Objective measurable evaluation criteria for assessing the results of the next iteration(s).

Towards the end of the construction phase the following artifacts must be produced and are additional exit criteria for the last iteration of the phase:

- a deployment plan, specifying as necessary:
  - production and packaging (e.g., making CD's and manuals)
  - pricing
  - roll out
  - support
  - training
  - transition strategy (e.g., an upgrade plan from an existing system)
- user documentation

3.3.4 Transition Phase

The transition phase is the phase where the product is put in the hands of its end users. It involves issues of marketing, packaging, installing, configuring, supporting the user community, making corrections, etc.
From a technical perspective the iterations continue with one or more releases (or deliveries), e.g. beta releases, general availability releases, bug fix or enhancement releases.
The phase is completed when the user community is satisfied with the product: formal acceptance for example in a contractual setting, or when all activities on this product are terminated. It is the point where some of the accumulated assets can be made reusable by the next cycle or by some other projects.

Entry criteria

- The product and artifacts of the previous iteration and in particular a software product sufficiently mature to be put into the hands of its users.

Exit criteria

- An update of some of the previous documents, as necessary, the plan being replaced by a "post-mortem" analysis of the performance of the project relative to its original and revised success criteria.
- A brief inventory of the organizations new assets as a result this cycle.

3.4 Process Workflows

3.4.1 Requirements Workflow

The job of an analyst is to identify the problem that an organization will address. He represents stakeholders by capturing, organizing and managing their expectations and requests for the system. To clearly define the right system, the analyst interprets stakeholder requests and articulates this information as requirements.
The following figure shows a typical requirements workflow. Each detail of the workflow represents an activity to perform to ensure effective requirements management.

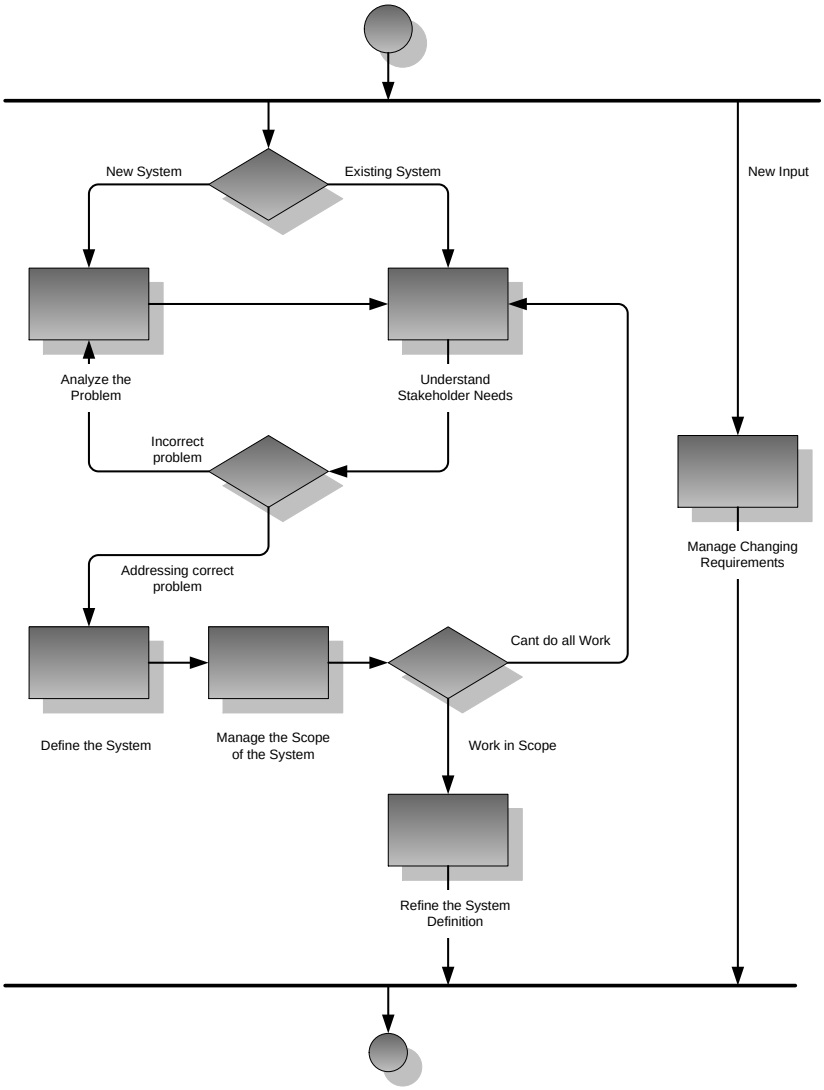

Figure 6 Requirements Workflow

The main requirements:

- Define the boundaries of the system.
- Provide a basis for planning the technical contents of each iteration.
- Provide a basis for estimating cost and time to develop the system.
- Define a user interface for the system, focusing on the needs and goals of the users.

Throughout this effort, the analyst must work with all stakeholders to clarify the problem space and solution requirements. Even as the development objectives and requirements change during the project, the job is to maintain communication with stakeholders and a shared understanding of the requirements.

3.4.2 Analyze and Design Workflow

The job of a developer is to review project requirements and to make decisions about the structural elements and interfaces of the system. As requirements change and new problems arise in each iteration, the developer must refine the system architecture.
The following figure shows a typical requirements workflow, which shows how developers verify that a design model fulfils system requirements and that it serves as a good basis for its implementation.

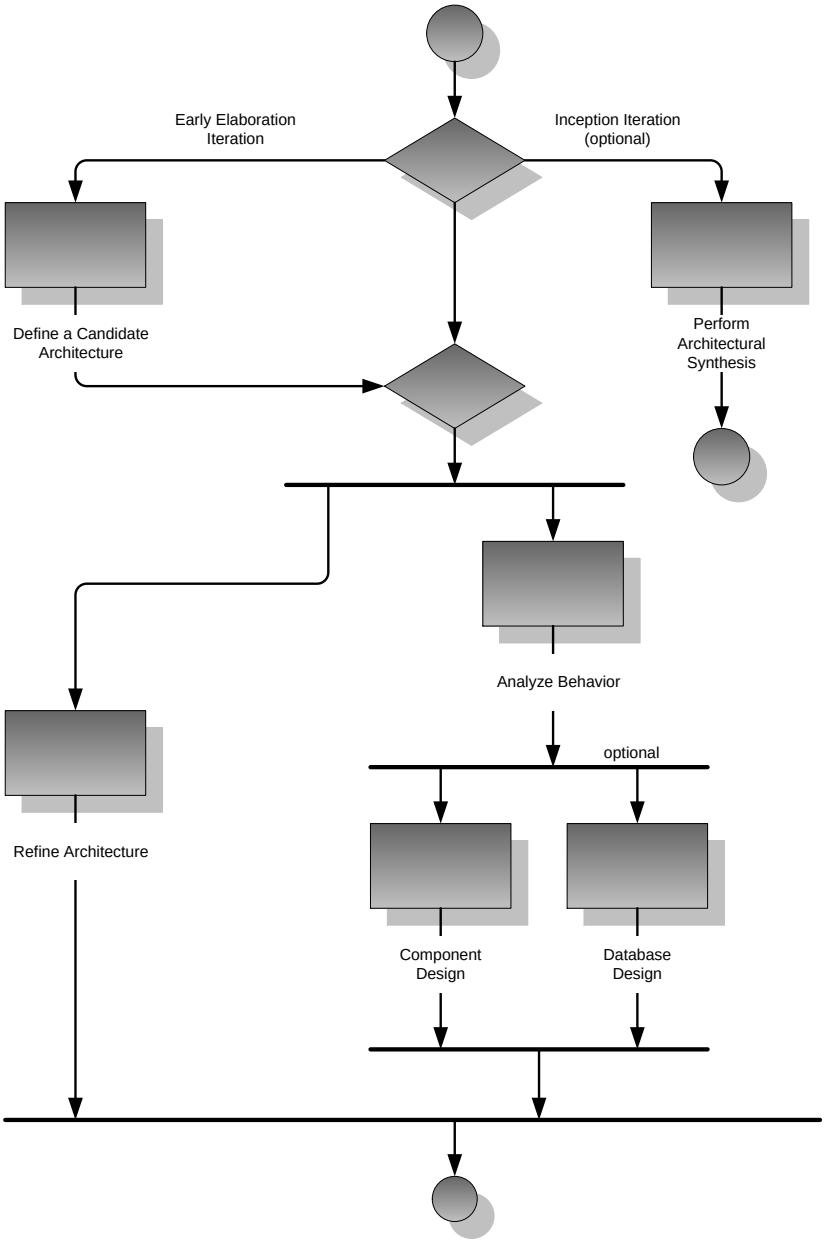

Figure 7 Analyze and Design Workflow

In the system design, the developer lay out the components of the architecture. These individual components are independent, replaceable parts of the system that have clearly-defined functions and interfaces. When the developer works with component-based architectures, the team can easily create new components. Team members can also reuse, or even customize, existing components from previous projects or commercially available sources.
The job of a developer is also to produce executable code that can be evaluated against the project requirements and the design during each iteration. As a responsible developer, the code will test before it release to other developers or to the testing group. This approach lets the developer to discover and respond to problems early enough to minimize their impact on the project.

3.4.3 Test Workflow

The job of a tester or a test team is to test releases in planned increments. During the complete project lifecycle, the tester manages and track use cases, requirements and tested code.
The following figure shows a typical test workflow. Each workflow detail represents the key skill needed to verify the proper integration of all components of the software, to verify that all requirements have been correctly implemented and to ensure that defects are addressed before releasing the software.

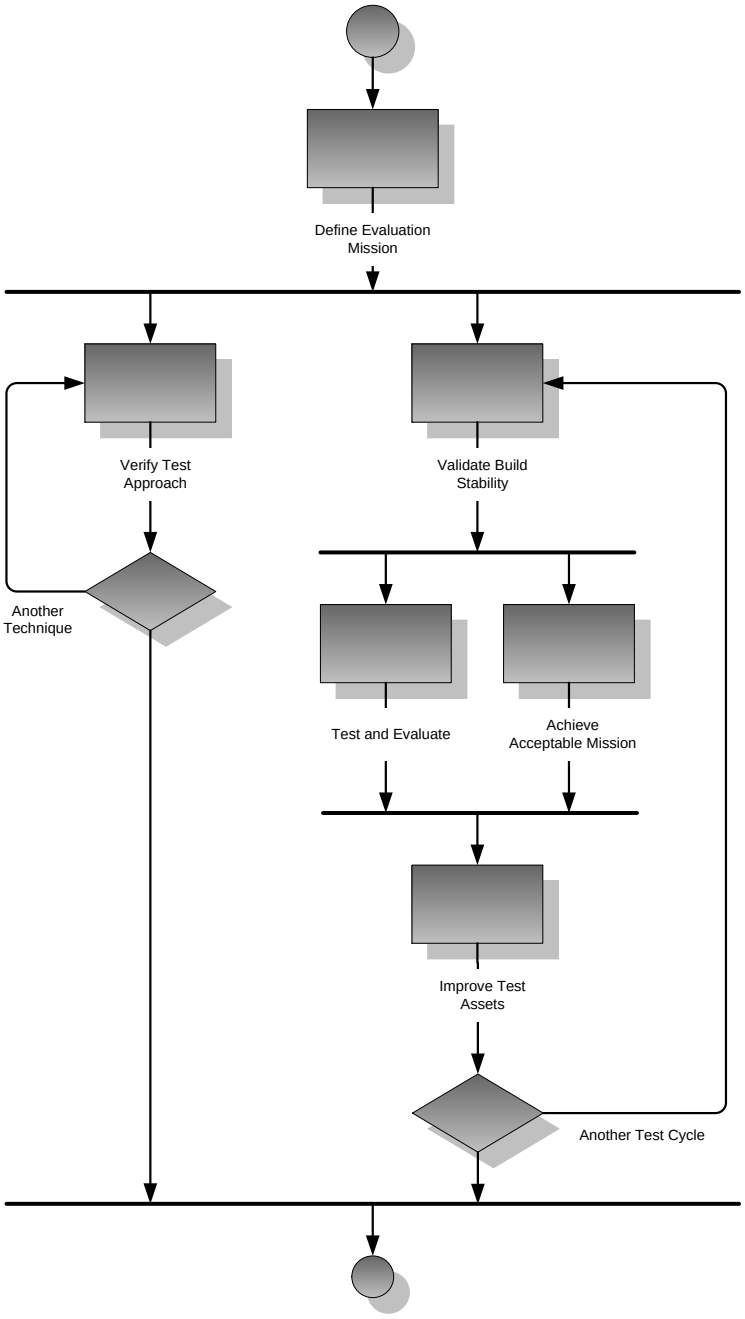

Figure 8 Test Workflow

Often, the tester retests code because of updated requirements or repaired defects. The complete team also runs regression tests on new builds to detect whether new bugs have appeared where they did not exist in previous builds.

3.4.4 Project Management Workflow

A project leader must identify and manage project risks, monitor the team progress and plan each iteration. The ongoing responsibilities are to assign and schedule work and to monitor the progress of the project.
The following workflow shows how project leaders plan an iterative project, its iterations and monitor progresses.

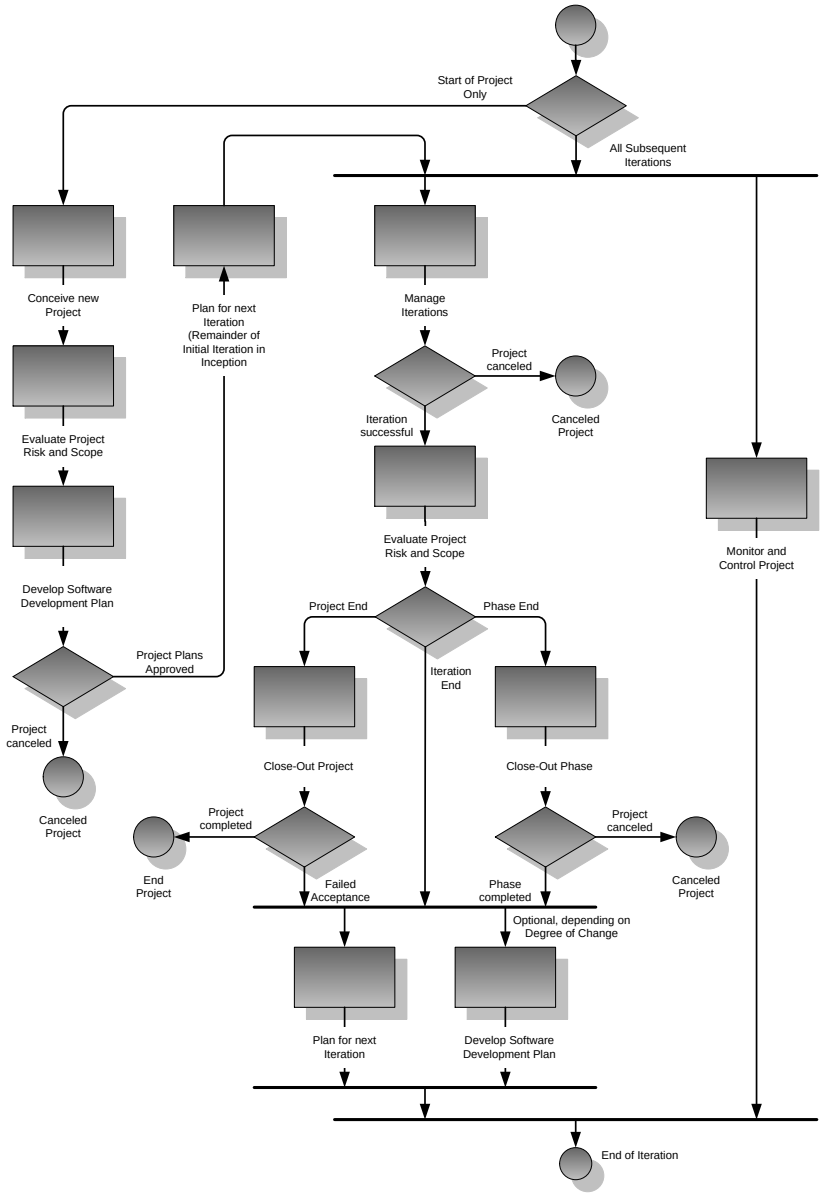

Figure 9 Project Management Workflow

Early in the development lifecycle, the project team identifies, implements and tests the most risky features and architectures.
Monitoring project progress involves collecting and assessing the latest metrics or status reports from each team member. Throughout development, the project leader analyzes project data to determine how well the team is meeting its objectives. He also uses this data to manage change and plan subsequent iterations.

In general, the art of project management involves balancing competing objectives, managing risk and overcoming constraints to successfully deliver a product that meets the needs of all stakeholders.

3.5 Process Artifacts

The Rational Unified Process® is not document-driven: its main artifact must remain at all time the software product itself. The documentation should remain lean and limited to the few documents that bring real value to the project from a management or technical point of view.

3.5.1 Management Artifacts

The management artifacts are not the product, but are used to drive or monitor the progress of the project, estimate the risks, adjust the resources, give visibility to the customer (in a contractual setting) or the investors.

- An Organizational Policy document, which is the codification of the organizations process; it contains an instance of this generic process.
- A Vision document, which describes the system level requirements, qualities and priorities.
- A Business Case document, describing the financial context, contract, projected Return On Investment (ROI), etc.
- A Development Plan document, which contains in particular the overall iteration plan and a plan for the current and upcoming iteration.
- An Evaluation Criteria document, containing the requirements, acceptance criteria and other specific technical objectives, which evolves from major milestone to major milestone. It contains also the iteration goals and acceptance levels.
- Release Description documents for each release.
- Deployment document, gathering additional information useful for transition, training, installation, sales, manufacturing and cut-over.
- Status Assessment documents, which contains periodic snapshots of project status, with metrics of progress, staffing, expenditure, results, critical risks and actions items.

3.5.2 Technical Artifacts

These artifacts are either the delivered goods: executable software and manuals, or the blueprints that were used to manufacture the delivered goods: software models, source code and other engineering information useful to understand and evolve the product.

- A User Manual, developed early in the life-cycle.
- Software documentation, preferably in the form of self-documenting source code and models (uses cases, class diagrams, process diagrams, etc.) captured and maintained with appropriate CASE tools.

- A Software Architecture document, extracted (abstracted) from the software documentation, describing the overall structure of the software, its decomposition in major elements: class categories, classes, processes, subsystems, the definition of critical interfaces and rationale for the key design decisions.

Depending on the type of project, this typical document set can be extended or contracted, some documents can be merged. The documents do not have to be paper documents - they can be spreadsheet, text-files, database, annotations in source code, hypertext documents, etc. - but the corresponding information source must be clearly identified, easily accessible and some of its history preserved.

3.5.3 Requirements

The Rational Unified Process® is not requirement-driven either. The requirements for the product evolve during a cycle and take different forms:

- The Business Case gives the main constraints, mostly in terms of resources that can be expended.
- The Vision document describes only the key requirements of the system from a users perspective and it evolves only slowly during the development cycle.
- The more detailed requirements are elaborated during the elaboration phase, in the form of use cases and scenarios and are refined incrementally throughout the construction phase, as the product and the users needs become better understood. These more detailed requirements are in the evaluation criteria document; they drive the definition of the contents of the construction and transition iterations and are referenced in the iteration plan.

4 Planning a RUP® Project

Although the project management discipline outlined in the Rational Unified Process® is often not fully appreciated by the development team, project planning is a critical activity for software development. Good planning helps the teamwork together to achieve a set of defined goals in a defined period of time.

4.1 Definition of Project Plan

One of the biggest problems managers face when dealing with a software development project is that by its very nature, the project is invisible and non-tactile. It is not like building a bridge, where everyone can see the progress that is being made. Because the physical result of the software development project - a running application - and its ongoing progress is not readily visible, it can be very difficult for the team to visualize and assess the project status. To deal with this invisibility, the primary practitioners on the project use abstraction. The architect has a UML model, the analyst has a requirements model (use cases), the tester a test plan. The project plan is the equivalent tool for the project manager. It provides an abstraction or model for the project manager to work with, share with the team and use to perform impact analysis.
In the modern software development environment, it is crucial to have a shared and dynamic vision of the project for the team to access and share.

A project plan performs these functions:

- Helps the manager plan the cash flow and schedule for the project.
- Communicates what is going to be delivered and when.
- Identifies which resources should be available and when they are required.
- Helps avoid clashes between competing resources on different activities.
- Helps the team know who is doing what on the project.
- Provides a basis for measuring progress and expenditures.
- Gives the planner some baseline to support replanning activities.
- Helps the customer and management to see what went wrong when a project runs aground.

A project plan has these key characteristics:

- The plan is target based - it identifies something that must be delivered on the project. If the plan is to be used as an aid to motivating the team toward a defined goal, it must provide clear targets for both the team and individuals to measure their performance against the plan.
- The plan enables the project manager to understand which team members are working on which tasks and what the dependencies are between those tasks.

- The plan provides many different views of the information, as required by different stakeholders (customers, team members and management). For example, it might offer a coarse-grain plan, an artifact plan, a delivery plan and a worker to-do list, among other things.
- The plan is measurable from a time perspective as well as a project delivery perspective. Often when asked about progress, a project manager can report how much time and money has been spent but cannot quantify how much of the system has been delivered. It is important for the project manager and the entire team to know the current state of the project, which key deliverables have been completed and which key deliverables are forthcoming.
- The plan is up to date. It is connected to the actual tasks being performed on the project and is the first place a project manager looks when assessing progress. If a project plan becomes secondary when assessing performance, it is not being used correctly.

4.2 Characteristics of a Project

A RUP® project has these two primary aspects that are important to the project plan: projects are iterative and the project progress is measured against clear milestones.

4.2.1 Iterative Development

The majority of the RUP® projects are, by definition, iterative. The RUP® is an incremental process whereby the overall project is broken down into phases and iterations. The iterations are risk driven - that is, oriented toward mitigating risks - and each one should deliver executable software that is demonstrable and testable against the projects requirements and use cases.
The project manager uses iteration plans to manage the project. Generally, work that falls outside of an iteration plan should not be undertaken.

An iteration plan:

- Provides a detailed description of the upcoming phase of work.
- Defines the worker roles involved, necessary activities and artifacts to be delivered in that iteration.
- Outlines a very clear set of measurement criteria by which progress can be assessed during the iteration and success can be measured at the end. Defines specific start/end dates and delivery dates.

4.2.2 Milestones

The RUP® identifies four phases for development projects: inception, elaboration, construction and transition. Each phase focuses the project team on a particular aspect of the project and has associated with it a number of milestones. These milestones help the project manager assess project progress and ensure that the project will deliver required features and will have quality built in.
In the context of iterative development, the milestones for a phase provide a focus for the iterations. Each iteration moves the project through certain milestones. For example, a iteration within the inception phase would be structured around the need to understand the scope of the project; the iteration(s) would provide the management framework for the team to explore the system boundary, implications of a possible solution and the size of that solution. The number of iterations would depend on how difficult it might be to define the scope of the project. If the scope were very hard to understand or could be grouped into easily defined pieces, more than one iteration might be needed. If, as is normally the case, it would take one clear piece of work to understand the scope, one iteration would be appropriate.
The milestones defined in the RUP® are of necessity quite general; the project manager will need to refine the milestones so they focus the team on the needs of the project in its particular organizational context. In addition, because the aim of a iteration is to mitigate risk, during an iteration the team will be resolving issues that apply not only to the focus of the phase but also to other disciplines, such as architecture, testing, change management or construction. The manager combines the iterative, risk-oriented approach with the refined milestones to determine the structure of the project plan.

4.3 Development of a Project Plan

For planning purposes, the RUP® clearly distinguishes between project planning and planning for a specific iteration.
Fundamentally, project planning involves developing a coarse-grained plan for the entire project, whereas iteration planning deals with developing a fine-grained plan for the specific upcoming iteration. In this section, the team discuss in detail how to develop a project plan.

4.3.1 Project Start Activities

At the start of a project, the manager needs to determine whether it is in the organizations best interests to engage in the project.
This basic question is often overlooked or taken for granted, but the exercise of answering it will result in the all-important two C's: Consensus and Commitment. The RUP® refers to this project planning stage - which consists of determining the economic viability of the project (developing a business case), making a start at identifying and assessing risks and initiating the project - as the Project Start.

Developing a Business Case
In developing a business case, the project manager documents the economic value of the proposed product. The artifact resulting from this activity is the instrument through which funding for the project is obtained and on which the key stakeholders agree. This artifact can be one page or one hundred pages. The important element here is to ensure that the manager have performed the due diligence as a manager to ensure that it is indeed in the companies best interests to undertake this project.
Keep in mind that the business case is often "make or break" for the product or project, so time spent laying it out is time well spent. Consensus by stakeholders on the economic fundamentals and market need for the product will be essential to gaining funding, resources and commitment from the organization. This commitment will ideally serve to drive the project development in the coming weeks or months through completion.
The RUP® recommends taking the following steps to develop the business case for the project:

- Describe the product and the need it fulfils.
- Define the business objectives and intended market for the product.
- Define the product or project objectives at a high level.
- Develop a financial forecast including projected revenues and costs for the project.
- Describe the project constraints that could potentially impact risk and cost.
- Describe the options that could impact the project success.

Identifying and Assessing Risks

Identifying and assessing project risks are an essential start-up task. The resulting artifacts will serve as the basis for risk mitigation and the development of iterations in the forthcoming elaboration and construction phases.
The RUP® recommends that project managers take these steps:

- Identify potential risks that would decrease the likelihood that the development team will be able to deliver the project with the right features, the specified level of quality, on time and within budget.
- Analyze and prioritize the risks by estimating the impact of each and the likelihood of its occurrence to determine the risk exposure for each risk.
- Identify risk avoidance strategies to reorganize the project to reduce or eliminate risks.
- Identify risk mitigation strategies.
- Identify risk contingency strategies.
- Revisit risks frequently within iterations and subsequent phases.

Initiating the Project
The Initiate Project activity is carried out after the projects business case is approved. This activity sets up the necessary executive management and project planning teams (if applicable) and also sets out the criteria that will be used to determine when the project has been successfully completed.
It consists of these steps:

- Assign a project review authority responsible for overseeing the project. For small projects, this authority can be a single person.
- Assign a project planning team, a group of project team members who will carry out the work of planning the project, maintaining the project plan and reporting on the ongoing project status.
- Approve project acceptance criteria - objective criteria that will be used by the customer or key stakeholders to determine when the artifacts delivered by the project are acceptable.

4.3.2 Project Organization and Staffing

Assuming that the project is considered viable after the start activities, the next activity will be to define an organizational structure for the project and to define staffing requirements based on effort estimates.

- Define the project organization based on the characteristics of the project and external constraints, such as existing organizational policy.
  The RUP® suggests defining the organizational structure of the project in terms of positions, teams, responsibilities and hierarchy.
- Define staffing requirements based on the effort estimates for the project, the desired schedule, the chosen organizational structure and mapping of roles.
  The RUP® recommends defining the numbers, type (skills, domain) and experience of staff required for the project. In addition, the

RUP® recommends adjusting the software development team from its baseline as the project moves through its development lifecycle. For instance, the project team will normally be heavy on management functions, particularly project management, in the inception phase. When the project moves into the elaboration stage, there will be more architectural staff. In construction, the development staff will be larger. Finally, when the project reaches the transition stage, the team should be heavy on QA or software assessment staff.

4.4 Compilation of a Software Development Plan

The next critical phase of developing a project plan is the creation of the Software Development Plan (SDP). The major activities of the SDP are defining the project structure and estimating the size of the overall project.

4.4.1 Project Structure

The RUP® has derived a frame estimate for project planning purposes of how time should be allocated among the phases.

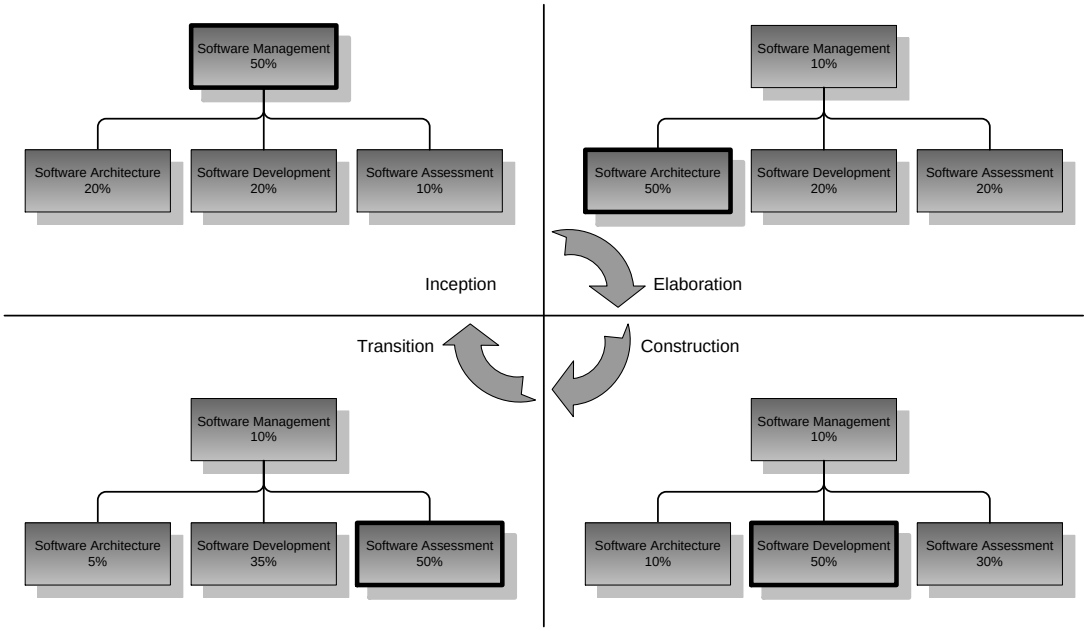

Figure 10 Rational Unified Process® Timeframe

Below are key questions whose answers will shed light on the number of iterations. Keep in mind that during the life of the project, the structure may change if phase milestones are not passed or if the project runs into problems.

Inception Phase

Question	Impact
Is the project going to deliver a business system as well as a computer system ? Is the project responsible for changing the business processes as well as building a computer system ?	Business process re-engineering is a very complex endeavor and the RUP® only provides techniques for business modeling, not for the supporting process. If, however, the business process is simple or well understood, it is possible that work on it may be undertaken in the inception phase. If this is the case, a more complex inception phase will be required.
Is there an existing business system ? If so, is it important to understand the existing business system ?	If there is a need to understand the existing business system as well as the new system, two iterations are needed - one focused on the existing system and the second on the new system.
Does everyone on the project have a good understanding of the business system that is being supported by the computer system ?	If there is not a good understanding on the part of a team, it is worth doing business modeling and increasing the length of the inception phase. Having a clear understanding of the problem, the team is trying to solve a crucial to a successful project.

Table 2 Key Questions for the Inception Phase

If the inception phase is considered complex (as determined by the questions above), the time allocation should be increased by 5% of overall project time. In addition to the relative size of the phase, the number of iterations needs to be defined. If there is a need to investigate an existing system, the number of iterations should be at least two. If the existing system does not need to be understood, the inception phase normally comprises just one iteration.

Elaboration Phase

Question	Impact
Is the system under consideration a new system ? Has the team built something similar to this system before ?	If there is no existing architecture, it will take time to create one. New systems make the elaboration phase more complex.
Is a pre-built framework going to be used for construction ?	The use of a pre-built framework provides a number of assumptions about the overall system architecture and really reduces the amount of effort required in the elaboration phase.
Is the technology to be deployed new to the team ?	If the technology is new to the team, there is often a steep learning curve associated with its adoption. This will lengthen the elaboration phase.
Does the system need to support concurrent threads of control or have to respond to asynchronous events ?	Additional activities are associated with real-time systems development and will lengthen the elaboration phase.

Table 3 Key Questions for the Elaboration Phase

If the elaboration phase is considered complex (as determined by the questions above), the time allocation should be increased by 5% of overall

project time. The number of iterations in the elaboration phase is determined by the technical architectural risks that are to be mitigated. Experience indicates that if the architecture is complex or the team is new to the technology to be deployed, at least two iterations will be needed during the elaboration phase.

Construction Phase

Question	Impact
Exists a separate integration test environment ?	If there is a separate integration environment, it may be possible to have multiple construction releases to this environment for integration testing. If there is no integration environment, the number of releases needs to be kept to a minimum.
Is the development team spread over many locations, so that members will be working in parallel ?	A development team that is spread over many locations increases the complexity of the build process and lengthens the construction phase.
Are the developers involved with the construction phase familiar with the approach and technology ?	If the team is new to the technology or methodology being applied, the construction phase might be difficult and will last longer.

Table 4 Key Questions for the Construction Phase

If the construction phase is considered complex (as determined by the questions above), the time allocation should be increased by 5% of overall project time. The number of iterations in the construction phase is determined by both the capability of the team to deliver incrementally and the underlying technical environment. If both the team and the environment are capable of supporting rapid incremental delivery, the number of iterations is defined by the logical pieces of functionality (collection of use cases) that can be delivered.

Transition Phase

Question	Impact
Is there a planning to run a beta program ?	If a beta program is planned, at least two transition iterations are needed.
Can the system incrementally delivered ? If so, how many increments?	Many business environments can not support a rollout that is undertaken incrementally. If the supporting business system must change or the technical environment is being replaced, the software product must be released in one big bang. It may be possible to determine deployment increments based on the geographical distribution of the business.

Table 5 Key Questions for the Transition Phase

If there are a large number of deployments (increments), the time allocation for the transition phase may need to be increased to 15% of total project time.

For a process phase overview example see the spreadsheet "RUP Process Phases".

4.5 Development of a Iteration Plan

Once the project manager has completed the coarse-grained project plan, an iteration plan develops at a more detailed level before every iteration. Each iteration in a phase moves the project toward the phase milestones and iteration plans are the mechanism by which this movement happens. An iteration plan defines a clear objective and associates with it the activities required to accomplish that objective. This section discusses how to develop iteration plans.

4.5.1 Determine Phases

The phase the team is in helps to determine the objective of a particular iteration.
For example, if the project is in the inception phase, the objectives will be as follows:

- Establishing the projects scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be in the product.
- Exhibiting and maybe demonstrating, at least one candidate architecture against some of the primary scenarios.
- Determining the critical use cases of the system, the primary scenarios of operation that will drive the major design tradeoffs.
- Estimating the overall cost and schedule for the entire project (and, in more detail, for the elaboration phase that will immediately follow).
- Estimating potential risks (the sources of unpredictability).
- Preparing the supporting environment for the project.

The factors defined earlier in the planning process will help the manager determine which of these objectives to focus on. Each of these objectives is associated with a deliverable.

4.5.2 Determine Deliverables

When the team understands the objectives of the phase, the team understands the deliverables for the iteration - that is, the things produced during that iteration. A deliverable may be something like a vision (of the system) or a series of use cases. Each deliverable has

associated with it a workflow template identifying the activities, resources and other deliverables that will be required to complete this deliverable. It also has associated with it an artifact, meaning the documentation that the project produces for that deliverable and that can be used to review the deliverable at the end of the iteration or workflow.
The following deliverables are samples:

- Business architecture - Describes the business processes and how those business processes are realized within the activity system being investigated. This deliverable is needed only if business modeling is being undertaken.
- Business case - Provides benefit and cost information about the product being built and defines success criteria for the project.
- Vision - Defines what the product does, the market it is aimed at and the main features of the product.
- Use case model - Defines the functional requirements of the system.
- Supplementary requirements - Defines the supplementary or nonfunctional requirements of the system.
- Use case - Describes a service provided by the system.
- User interface prototype - Simulates the user interface, as defined and testable by users. This is very important if the system has a very complex GUI or has a series of nonfunctional requirements relating to usability.
- Analysis model - Identifies key abstractions that make up the system.
- Subsystem design - Describes the design of subsystems of the analysis model, which will consist of a series of components.
- Risk list - Describes the risks to be mitigated during each iteration.
- Component - Consists of an executable unit of code to be deployed in the executable system.
- Build - Groups a series of components into an entire system.
- Release - Consists of the particular collection of activities that defines a particular release.
- Functional test - Tests the functionality needed to meet a particular requirement.
- Performance test - Tests the performance of the system.
- Test environment - Sets up the test environment for a particular iteration.
- Defect/Enhancement - Outlines the process for resolving defects or performing enhancements in response to user feedback on the other deliverables.
- Development environment - Sets up the development environment and manages changes to this environment.

4.5.3 Selection of the Appropriate Workflow Template

For each deliverable, there is an associated workflow template in the RUP® that identifies the activities required to realize this deliverable. An activity includes roles, artifacts and guidance (help steps). Certain activities may not be appropriate for the particular project or iteration and these can be omitted. The amount of time allowed for activities will depend on the total time allocated for this iteration in the project plan.

4.5.4 Assocation of Resources with Activities

Resources need to be associated with each particular activity in the iteration plan. The workflow template provides hints as to which resources are required by defining the roles that need to be played in each activity. The planner must fill the roles with actual workers.

4.5.5 Definition of Monitoring and Control Processes

The project manager needs monitoring and control processes that check vital signs of the project with regard to the software development plan. Typically, the project manager will be concerned with indicators that apply to the projects scope of work, progress, budget, quality and risks. The RUP® suggests that project managers do the following:

- Define project indicators to alert the project manager to instigate corrective actions as required.
- Define sources for project indicators.
- Define and communicate a procedure and reporting frequency for project team members to report their status.
- Define thresholds for the project indicators.
- Define a procedure for project status reporting.

4.5.6 Assessment of Iterations

After completing an iteration, the project manager and the development team should assess the success or failure of the iteration and capture lessons learned, so they can be applied to modify aspects of the project or to improve the process. This very important step will also help ensure that future projects will reflect the lessons learned in the iteration.
Project managers should do the following:

- Collect metrics and progress information on the project so that actions can be taken that may involve re-planning, retooling, training, or process reorganizing, team and tools.

- Assess the results of the iteration as compared with expected results, in terms of functionality, performance, capacity and quality measures.
- Examine the evaluation criteria to ensure that the goals are not set too high or too low, that the requirements are still valid and that the features are still economically justified.
- Use the results of the assessment to generate change requests for the vision, the risk list, the project plan, subsequent iteration plans or the development case and requirements.

5 Overview of Available Tools

The following table shows which tools are included in the available editions of the Rational Suite.

		Project Leaders / Managers	Analysts	Developers				Testers	All Roles	Web Teams
		Team Unifying Platform	Analyst Studio	DevelopmentStudio		DevelopmentStudio RealTime Edition		Test Studio	Enterprise	Content Studio
	Tool	Windows	Windows	Windows	UNIX	Windows	UNIX	Windows	Windows	Windows
Team Unifying Platform	Rational Unified Process	●	●	●	●	●	●	●	●	●
	Rational RequisitePro	●	●	●	● (Web)	●	● (Web)	●	●	●
	Rational ClearQuest	●	●	●	● (Web)	●	● (Web)	●	●	●
	Rational SoDA	● (W)	● (W)	● (W)	● (F)	● (W)	● (F)	● (W)	● (W)	● (W)
	Rational ClearCase LT	●	●	●	●	●	●	●	●	●
	Rational TestManager	●	●	●	●	●	●	●	●	●
	Rational ProjectConsole	●	●	●		●		●	●	●
Rational Rose			● (M)	● (E)	● (U)	● (RT)	● (RT/U)		● (E)	
Rational PureCoverage				●	●	●	●	●	●	
Rational Purify				●	●	●	●	●	●	
Rational Quantify				●	●	●	●	●	●	
Rational Robot								●	●	
Rational TestFactory								●	●	
Rational Process Workbench									●	
Rational NetDeploy										●
Rational SiteLoad										●

M Professional Data Modeler Edition
E Enterprise Edition
U UNIX Edition
RT RealTime Edition
W Microsoft Word
F Adobe FrameMaker
Web Access also through the Web Interface

Table 6 Rational Tools Overview

5.1 Rational Suite Team Unifying Platform

Rational Suite Team Unifying Platform is a suite of tools that is included in every edition. It provides best practices and integrated tools for managing change, building quality and communicating results from requirements to release.

The Team Unifying Platform is designed for project members who need access to common project artifacts, but do not need any of the optimized, role-specific tools found in the other editions. Project and program managers, project administrators and development managers use the tools in this edition.

The Team Unifying Platform Suite edition contains the following tools:

5.1.1 Rational Unified Process®

The Rational Unified Process® is an online collection of software best practices that guide a team through the software development process. It provides guidelines, templates and tool mentors for each phase of the development lifecycle.

Rational Unified Process® platforms: Windows, UNIX

5.1.2 Rational RequisitePro®

Rational RequisitePro® helps to organize, prioritize, track and control changing project requirements. RequisitePro® comes with the RequisiteWeb® interface which lets users to access, create and manage requirements from a Web browser.

RequisitePro® platforms: Windows
RequisiteWeb® platforms: Windows, UNIX

5.1.3 Rational ClearQuest®

Rational ClearQuest® manages change activity associated with software development, including enhancement requests, defect reports and documentation modifications.
The ClearQuest® Web interface lets users to perform all major ClearQuest® operations, such as submitting records, finding records, creating or editing queries and reports, and creating shortcuts, from a Web browser.
ClearQuest® MultiSite lets users to share information across a geographically distributed team.

ClearQuest® platforms: Windows, UNIX (a Windows workstation must configured as administrator for the ClearQuest® repository)
ClearQuest® Web Interface platforms: Windows, UNIX

5.1.4 Rational ClearCase® LT

Rational ClearCase® LT provides software configuration management and a built-in process to track changes to all software project assets, including requirements, visual models and code. It also provides a Web interface that lets users perform all major ClearCase® LT operations. Rational ClearCase® LT supports Unified Change Management, the Rational best-practices-process for managing change and controlling workflow.

ClearCase® LT platforms: Windows, UNIX

5.1.5 Rational SoDA®

Rational SoDA® automatically generates project documents by extracting information from files produced during project development, including source code and files produced by Rational tools. SoDA® uses templates, either predefined or ones that the user customize, to format the information.

SoDA® platforms: Windows (Microsoft Word), UNIX (Adobe FrameMaker)

5.1.6 Rational TestManager

Rational TestManager helps to create real-world functional and multiuser tests to determine the performance and reliability of Web, multitier and database applications. TestManager tracks the number of tests that have been planned, scripted and run; which requirements have been covered; and the number of tests that have passed and failed. TestManager gives a team the information it needs to objectively assess project status and create reports to communicate these findings to project stakeholders.

TestManager platforms: Windows, UNIX

5.1.7 Rational ProjectConsole®

Rational ProjectConsole® helps to track project metrics by automatically generating charts and gauges from data produced during software development. ProjectConsole® is integrated with Microsoft Project so that the user can create a centralized project plan. ProjectConsole® helps to organize project artifacts on a central Web site so all team members can view them.

ProjectConsole® platforms: Windows

5.1.8 Rational ContentStudio®

Rational ContentStudio® offers distributed authoring capabilities, allowing Web content contributors to add and update content using templates. These templates allow the Web development team to separate the text and editable portions of a site from the design elements, so content contributors can modify Web material while maintaining a consistent look and feel of the Web site.

ContentStudio® platforms: Windows

5.1.9 Rational NetDeploy®

Rational NetDeploy® simultaneously delivers all changes to related code and content to multiple staging or production servers. NetDeploy® lets to schedule one-time, recurring or dependency-based deployments. It can also automatically identify groups of files (code and content) that need to be deployed together.

NetDeploy® platforms: Windows

5.1.10 Rational SiteLoad®

Rational SiteLoad® is a testing tool that simulates internet traffic and provides developers with precise real-time information on Web site performance.

SiteLoad® platforms: Windows

5.2 Rational Suite AnalystStudio®

Rational Suite AnalystStudio® is customized for team members who gather and manage project requirements.

AnalystStudio® platforms: Windows

5.3 Rational Suite DevelopmentStudio®

Rational Suite DevelopmentStudio® is customized for software architects, designers and developers. These team members use DevelopmentStudio® to design, evaluate, implement and test software architecture and applications.

DevelopmentStudio® platforms: Windows, UNIX

5.4 Rational Suite DevelopmentStudio®-RealTime

Rational Suite DevelopmentStudio®-RealTime is customized for software architects, designers and developers of real-time embedded software. This suite provides an integrated set of development and testing tools to optimize the development of complex software for devices such as cell phones and pagers and infrastructure software for the routers and hubs that connect and power the Internet.

DevelopmentStudio®-RealTime platforms: Windows, UNIX

5.5 Rational Suite TestStudio®

Rational Suite TestStudio® is designed for team members who are responsible for software quality assurance, functional testing, performance testing and load testing.

TestStudio® platforms: Windows

The Rational Suite TestStudio® edition contains the following tools:

5.5.1 Rational Robot

Rational Robot determines whether the system meets its requirements by testing how it responds to a user-driven scenario. With the interface of Robot, the user can record a test and insert verification points to monitor expected behavior. Also replay functions to test as often a developer need are integrated.

Robot platforms: Windows

5.5.2 Rational TestFactory®

Rational TestFactory® automatically generates tests that pinpoint severe defects: the places where the application crashes, hangs or behaves in unexpected ways. It also generates test scripts that exercise the maximum amount of code using the least number of steps. TestFactory® stores the test scripts, results and defect scripts in a project that it shares with Robot and other Rational testing tools. A team can generate coverage and progress reports from test results in this project. Robot can later rerun TestFactory® scripts to ensure that all tests are repeatable.

TestFactory® platforms: Windows

5.6 Rational Rose®

Rational Rose® helps to visualize, specify, construct and document the structure and behavior of the systems architecture. Rose® can provide a visual overview of the system using the Unified Modeling Language (UML), the industry-standard language for visualizing and documenting software systems.
Using visual models helps manage system complexity because the manager can see the "big picture."
Rational Rose® unifies the team by helping to create such models so that all stakeholders gain perspective and share a common understanding of the project goals, path and expected deliverables.
Modeling with Rose® is also an effective way to continuously communicate change and the impact of change throughout the development lifecycle.

The Rational Rose® edition contains the following tools:

5.6.1 Rational QualityArchitect®

Rational QualityArchitect® automates the mechanical aspects of test code creation by generating test code from visual models. This lets developers automatically generate component tests and build stubs and drivers before an application is complete. This feature helps to reduce project risk because the team can test early and often, determining how a potential system architecture meets functional and performance requirements before developing the design further.
Enterprise JavaBeans, COM, COM+ and DCOM models are supported.

QualityArchitect® platforms: Windows

5.6.2 Rational Purify®

Rational Purify® checks every active C++ and Java component in a program for run-time errors and memory leaks, the most difficult errors to find. They are the most important to correct because they often remain undetected until triggered by some random event. A program can appear to work correctly for a long time before these types of errors are discovered.

Purify® platforms: Windows, UNIX (C++ only)

5.6.3 Rational PureCoverage®

Rational PureCoverage® provides a report of each line in the code that has been run. This information lets determine if tests have actually run the lines of code that were intended to be tested.

PureCoverage® platforms: Windows

5.6.4 Rational Quantify®

Rational Quantify® detects performance bottlenecks, which are places where the code is running inefficiently. It pinpoints where the application is spending its time and helps to discover why a specific function is particularly slow. Quantify® helps to improve system performance so that the team can deliver efficient software.

Quantify® platforms: Windows

6 Usage of Tools

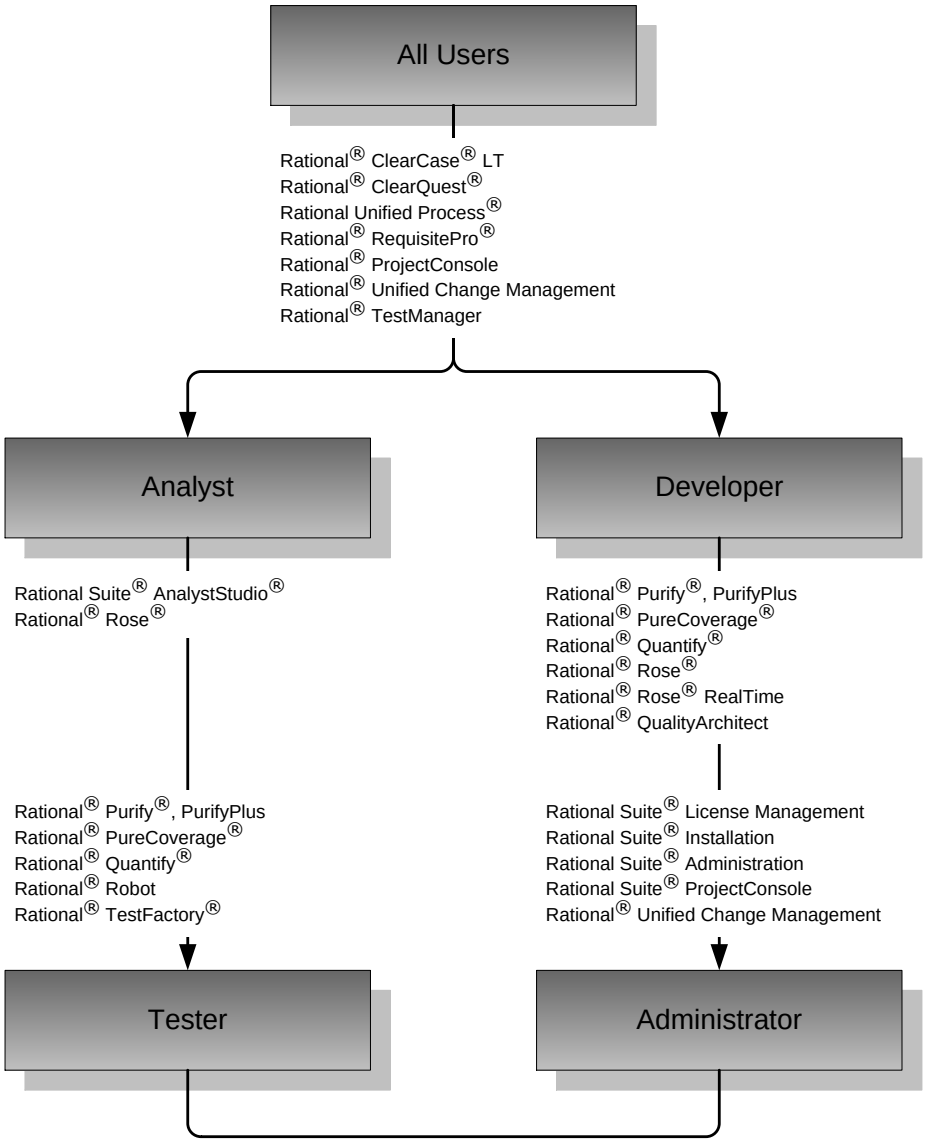

Figure 11 Rational Tools Roadmap

6.1 Analyst

6.1.1 Definition of Requirements

Rational RequisitePro® enhances team communication by helping analysts capture, manage and articulate requirements in a form accessible to all stakeholders.

- An analyst use Microsoft Word to document project requirements.
- Throughout the software development lifecycle, the analyst use requirements to communicate what to build and test. The requirements form the foundation of the system definition by defining the product vision and describing the systems features, functionality and attributes.
- The analyst store and track this information in the RequisitePro® database to optimize requirements management. The database assign attributes to requirements for organization, prioritization and change history. These attributes can be traced to related requirements so that the analyst can quickly assess the impact of any changes on requirements as development evolves.
- Because every team member must share a common understanding of project goals and objectives, the analyst keeps the team up to date on requirements activities.
- RequisitePro® makes this easy by providing Rational RequisiteWeb®. This Web-based tool lets UNIX users and other team members who do not have RequisitePro® on their desktops to create, review and update requirements.

6.1.2 Managing Changes

Rational ClearCase® LT is a configuration management solution for small project teams. This tool helps the team to manage changing artifacts, such as Web content, code, visual models and test assets as the system evolves.

- An analyst use ClearCase® LT to manage changes to project requirements. The tool also associates requirements with releases and other project assets.
- The analyst also uses ClearCase® LT to archive requirements. It tracks changes to all project files, allowing team members to work in parallel and continuously integrate their changes to the project baseline.
- ClearQuest® encourages collaboration by tracking new feature, enhancement or change requests from team members and other stakeholders. The ClearQuest® Web interface ensures that UNIX user and other team members who do not have ClearQuest® on their desktops can participate in this collaboration.

- With ClearQuest®, the team and other stakeholders can evaluate requests, determine their impact on the system and when applicable, validate the changes.
- To establish how change requests fit into the structure of features and main requirements, the analyst can link requests to an existing or new project requirement in RequisitePro®.
- ClearQuest® MultiSite lets geographically distributed teams replicate a centralized database at each remote site and then synchronize the changes between sites.

6.1.3 Team Communication

Rational Rose® helps the analyst to visualize, specify, construct and document the structure and behavior of the system architecture.

- With Rose®, the analyst can provide a visual overview of the system using the Unified Modeling Language (UML). Using visual models helps manage system complexity because the project team can see the "big picture."
- Rose® unifies the team by helping the analyst to create such models so that all stakeholders gain perspective and share a common understanding of the project goals, path and expected deliverables.
- Modeling with Rose® is also an effective way to continuously communicate change and the impact of change throughout the development lifecycle.

6.1.4 Progress Measuring and Project Reports

Rational ProjectConsole® extracts information from data produced during software development by tools like RequisitePro® and ClearQuest®.

- ProjectConsole® automatically generates pre-formatted or customized charts and graphs depicting the collected metrics. These metrics can share the project status information with members of the team.
- ProjectConsole® also analyzes data in a single view that reflects data collected from multiple sources.

Rational SoDA® generates up-to-date project reports of data extracted from one or more tools in Rational Suite.

- SoDA® can work with one tool, such as RequisitePro® or combine information from more than one tool, such as Rational RequisitePro® and ClearQuest®.
- SoDA® offers reporting features that provide templates in either Microsoft Word for Windows or Adobe FrameMaker.

6.1.5 System Test

Rational TestManager lets testers manage test planning, design, development, execution and analysis and share these assets with team members throughout the development lifecycle.

- With TestManager the analyst can build test assets from use cases.

6.2 Developer

6.2.1 Validation of Requirements

Rational RequisitePro® provides up-to-date requirements data in a form accessible to all stakeholders. This access is provided with RequisiteWeb®, a Web interface that lets all team members create, review and update requirements.

- When the developer see additions or changes to requirements in RequisitePro®, he can incorporate these changes into the project Rose® models.
- With this changes the developer gain an understanding of the impact these changes have on the system.
- The Developer can also communicate these changes and impacts to team members, e.g. analysts, developers, testers, project leaders and other stakeholders.

6.2.2 Managing and Validation of Changes

Rational ClearQuest® tracks features, enhancements or change requests from team members and other stakeholders in a form accessible to everyone.

- The ClearQuest® Web interface ensures that UNIX users and other team members who do not have ClearQuest® on their desktops can create and review project data and provide feedback.
- The developer use Rose® models to visualize the effects of these requests on the system architecture.

Rational ClearCase® LT helps the developer to control changes to source code and other project items, such as RequisitePro® databases and Rose® model files.

- ClearCase® LT tracks changes to every file and directory, maintaining histories of source code, binaries, executables, documentation, test suites, libraries and user-defined objects.
- The developer can use ClearCase® LT from a development environment, e.g. Microsoft Visual Studio, Visual Age for Java, as well as Microsoft Word, FrontPage, Visual InterDev and PowerBuilder.
- By organizing an effective development structure, ClearCase® LT helps software teams to accelerate build and release cycles.

Rational ProjectConsole® provides Web access to the projects current artifacts, metrics and best practices.

- ProjectConsole® automatically collects metrics at regular and user-specified intervals from Rational tools and Microsoft Project.
- These metrics are graphically displayed in personalized views on the project Web site. This reporting mechanism provides accurate, objective and up-to-date information to team members about the status and trends.

Rational SoDA® lets the developer to generate up-to-date project reports for the entire team by extracting data from one or more tools.

- SoDA® can work with one Rational tool or combine information from more than one tool.
- Its reporting features provide templates in either Microsoft Word or Adobe FrameMaker.

6.2.3 Team Communication

Rational Rose® helps the developer to visualize, specify, construct and document the structure and behavior of the system architecture.

- With Rose®, the developer can provide a visual overview of the system using the Unified Modeling Language (UML).
- Rose® unifies the team by helping the developer to create high-quality architecture models that all team members can share, test and revise.
- During design and code reviews, team members use project models to assess the ramifications of changes they want to make to the code.

6.2.4 Code and Model Implementation and Consistency

The developer can accelerate coding by generating code frameworks from models developed in Rational Rose®. This process is called "Forward Engineering". Rose® supports many languages, including Visual Basic, Visual C++, ANSI C++, Java and IDE's including Visual Age for Java, Visual Studio, HP Workbench and Sun Workshop.

After the developer modifies the code, he can use Rose® to bring code changes into the model. This process, "Reverse Engineering", ensures that the model and code remain consistent throughout the project.

6.2.5 System Test

A developer must test the code as soon as he implements it. Rational Suite provides testing tools to use as soon as the developer have a working program, allowing to test all dimensions of quality with automated debugging, performance testing and verification of code coverage.

Rational QualityArchitect® is a feature that automates the mechanical aspects of test code creation by generating test code from visual models.

- This lets developers automatically generate component tests and build stubs and drivers before an application is complete.
- This feature helps to reduce project risk because the team can test early and often, determining how a potential system architecture meets functional and performance requirements before developing the design further.
- Enterprise JavaBeans, COM, COM+ and DCOM models are supported in this feature.

Rational Purify® checks every active C++ and Java component in a program for run-time errors and memory leaks.

Rational PureCoverage® provides a report of each line in the code that has been run. This information lets the developer to determine if the tests have actually run the lines of code that were intended to be tested.

Rational Quantify® detects performance bottlenecks, which are places where the code is running inefficiently. It pinpoints where the application is spending its time and helps the developer to discover why a specific function is particularly slow. Quantify® helps to improve system performance so that a developer can deliver efficient software.

Rational TestManager helps the developer to keep track of the number of tests that have been planned, implemented and run.

- It also helps to track which requirements or Rose® model elements have been covered in testing and the number of tests that have passed and failed.
- Team members can use TestManager to evaluate how well they are meeting project requirements from early in the development lifecycle and communicate these findings to project stakeholders.

6.3 Tester

6.3.1 Team Communication

Rational RequisitePro® provides current, accurate information about requirements. RequisitePro® indicates to a tester that there are new or revised requirements that need to be tested.

- The tester uses the requirements and related artifacts to create a test plan and track testing progress.
- The RequisitePro® Web interface, RequisiteWeb®, lets team members and customers who do not have RequisitePro® on their desktops create, review and update requirements.

Rational ProjectConsole® unifies the team by allowing all contributors Web access to current project artifacts and metrics.

- This tool extracts information from data produced during software development and automatically generates charts and gauges, either predefined ones or ones that the user customizes.
- These metrics are graphically displayed in personalized views on the project Web site, providing accurate, objective and up-to-date information about the project status and trends.
- ProjectConsole® also lets the tester to analyze data in a single view that has been collected from more than one source.

Rational SoDA® extracts information from one or more tools and combines this information into reports about the project.

- The team can evaluate test results against requirements data from RequisitePro®.
- SoDA® offers reporting features that provide templates in either Microsoft Word or Adobe FrameMaker.

6.3.2 Progress Measuring

Rational TestManager lets the tester to use all types of project artifacts to plan, design and run tests.

- Requirements, visual models and source code are some of the test inputs that the tester can use to create a test plan, so that all aspects of the system can be tested, including product features, system architecture and code.
- With TestManager, test inputs are used to create specific test cases. A test case describes a testable and verifiable behavior in a system. It can also describe the extent to which the tester will test an area of the application.
- The tester can run multiuser performance tests for Web, multitier and database applications. Using simple point-and-click operations, the tester can create usage scenarios that simulate conditions in the system while it is being run by thousands of users.

- When the original artifact changes, the tester are automatically prompted to re-evaluate the related test asset. Then he can decide to update the test artifact, if necessary.
- The TestManager can verify whether the tester has tested all requirements and whether the test cases based on these requirements pass the tests. This relationship between project assets lets he tests for quality early in the development lifecycle, often beginning with product features and continuing through implementation and release.

6.3.3 System Test

Rational ClearQuest® tracks the defects that are found in the software project and provides a description, as well as other details, about the bugs.

- The ClearQuest® Web interface is a Web-based version of ClearQuest® that lets all team members create, review and update defects and change requests from any platform.
- Team members are able to access and create records, queries and reports on areas of interest specific through the Web module as they would through the client version.

Rational TestFactory® automatically generates tests that pinpoint severe defects: the places where the application crashes, hangs or behaves in unexpected ways.

- TestFactory® generates test scripts that exercise the maximum amount of code using the least number of steps.
- TestFactory® stores the test scripts, results and defect scripts in a project that it shares with Robot and other Rational testing tools. The team can generate coverage and progress reports from test results in this project.
- TestFactory® generates its own tests, the tester can start reliability testing early in the development process without having to budget more time to develop and run these tests.

Rational Robot determines whether the system meets its requirements by testing how it responds to a user-driven scenario.

- The tester can record a test and insert verification points to monitor expected behavior.
- The tester can also replay the test as often as he need.
- After the test, the tester can view the results and the complete details of any failures: what test was running, what type of failure occurred, where it occurred and which verification point failed.

Rational Purify® checks every active C++ and Java component in a program for run-time errors and memory leaks.

Rational Quantify® detects performance bottlenecks, highlighting places where the code is running inefficiently. It pinpoints where the application is spending its time and why a specific function is particularly slow.

Rational PureCoverage® highlights untested areas of the system so that the tester can build a more comprehensive set of tests.

Links

Rational® Software provides support for enterprise application development on J2EE, .NET, Linux and other platforms to support application deployment on the customer hardware and software platforms. The tools also are used to build technical software, commercial software products and software for embedded devices and real-time systems, such as pagers, cell phones, medical devices, air traffic control systems and government defense systems.
In 2002 the company was assumed by IBM.

Homepage: http://www-306.ibm.com/software/rational
developerWorks: http://www-136.ibm.com/developerworks/rational
FAQ: http://www-306.ibm.com/software/awdtools/rup/faq

References

Books:

Whitepapers: *Quality Management*, Certitude GmbH, Munich, February 2004

The Ten Essentials of RUP® - The Essence of an Effective Development Process, Leslee Probasco, Rational Software Corporation, Cupertino, 2000

Planning A Project with RUP®, David West, Rational Software Corporation, Cupertino

Articles: *A Rational Development Process*, Phillippe Kruchten, Crosstalk Magazine, July 1996

Trademarks and other Acknowledgements

Glossary

Artifact	Any document or software other than the software product itself.
Baseline	A release that is subject to change management and configuration control
Construction	The third phase of the process, where the software is brought from an executable architectural baseline to the point where it is ready to be transitioned to its user community.
Cycle	One complete pass through the 4 phases: inception-elaboration-construction-transition. The span of time between the beginning of the inception phase and the end of the transition phase.
Elaboration	The second phase of the process where the product vision and its architecture are defined.
Evolution	The life of the software after its initial development cycle; any subsequent cycle, where the product evolve.
Generation	The result of one software development cycle.
Inception	The first phase of the process, where the seed-idea, RFP, previous generation-is brought up to the point of being (at least internally) founded to enter into the elaboration phase.
Iteration	A distinct sequence of activities with a base lined plan and evaluation criteria.
Milestone	An event held to formally initiate and conclude an iteration.
Phase	The span of time between two major milestones of the process where a well defined set of objectives are met, artifacts are completed and decisions are made to move or not into the next phase.
Product	The software that is the result of the development and some of the associated artifacts (documentation, release medium, training).
Prototype	A release which is not necessarily subjected to change management and configuration control.
Release	A subset of the end-product which is the object of evaluation at a major milestone.
Risk	An ongoing or upcoming concern which has a significant probability of adversely affecting the success of major milestones
Transition	The fourth phase of the process where the software is turned into the hands of the user community.
Vision	The users view of the product to be developed.