Adopting RUP in a COTS implementation project Part I: Getting started

by Ronald Chan
Consultant
Rational Software Singapore
IBM Software Group

With the rising popularity of commercial off-the-shelf (COTS) systems, many organizations are beginning to acquire and integrate such systems into their enterprise IT solutions. Most organizations new to COTS implementation projects do not know how to plan or manage such projects. Serge Charbonneau's article in the March 2003 issue of The Rational Edge *looked at how the best practices embodied in the Rational Unified Process,® or RUP,® product can help in planning and managing a COTS implementation project. In this new article series, we will follow an actual case study for such a project. We will explore the COTS evaluation process, the formulation of a project vision, iteration planning, allocation of roles and responsibilities, the impact of contractual payment, and risk management within the project team.*

In Part I, we will focus on the early stages of an implementation, assuming that our readers have a fundamental understanding of RUP[1]. In many of today's large corporations, buying and integrating commercial off-the-shelf (COTS) systems as part of an enterprise IT solution has become more the norm than the exception. Business organizations, especially those with small to moderate-sized in-house IT support groups, typically purchase such "package solutions" and customize them to meet their business needs rather than building software solutions from the ground up.

Implementing these solutions often involves a third-party contractor (i.e., a *systems integrator/consultant* from either the COTS vendor or one of its partners) that provides the expertise needed to customize and integrate the COTS solution into the operational environment.

Although many project teams, recognize the benefits of using RUP for these implementations, they struggle for ways to apply RUP disciplines at an appropriate level of detail for all groups involved in the project. These may include the following:

- **In-house developers** who will maintain the COTS system.
- **Third-party contractors** who will implement the COTS solution.
- **Customers/end users** who will supply the requirements and eventually use the system.

Why use RUP?

COTS implementations, like those for systems developed from scratch, need complete and rigorous software engineering disciplines and best practices. Although implementing individual functions within a COTS system might require less design work, the overall system, with all its wrappers and integration "glue"[2] code, needs the benefits that best practices bring.

For example, iterative requirements management -- a RUP best practice -- coupled with close and continuous communication between the in-house developer team, contractor team, and customers/end users, is critical for implementing COTS systems. Customers/end users must understand their requirements well enough to make informed decisions and tradeoffs when choosing COTS workflows. These may be based on an inherent operational business model that does not match the real needs of the customers/end users. Therefore, for good requirements management, the customers/end users must be willing and able to prioritize their requirements and iteratively revisit their assessment as needed, as the team gains a deeper insight into the COTS capabilities.

Another RUP best practice involves managing the complexities of releases, patches, and upgrades with a configuration management process. In addition, this should be coupled with an effective defect tracking process. Whenever new releases, patches, and upgrades of COTS products are introduced into the system, regression testing is performed to ensure their stability and compatibility with the rest of the system; modifications to glue code, wrappers, and interfaces may be required, and the potential impact on the schedule must be considered. Therefore, projects commonly decide to stick to a particular baselined version of the COTS package (and its components) from Inception to delivery, to ensure stability. However, such a decision has to be made with a good understanding of the impact the "no upgrade until delivered" decision might have. For example, will the COTS still be upgradeable or supported if it missed more than two consecutive releases?

Typically, a project must also consider the following issues regarding changes:

- For each patch of a release, which sets of application programmable interfaces (APIs) are affected?

- For each release/upgrade, are there any changes to the COTS database structure or file content?
- Are releases, patches, upgrades, and configurations consistent across the organization?
- Are defect tracking and feedback mechanisms available as the COTS system stabilizes?

In short, the rigor of the RUP disciplines, executed within the framework of RUP's iterative, risk-driven approach, is just as necessary for implementing COTS systems as it is for implementing internally produced systems.[3] As Barry Boehm and Chris Abts put it:

> COTS-based projects cannot make blanket assumptions about system requirements, or embrace traditional process models. One mistake developers can make is to use the waterfall model on a COTS integration project.... Use risk-driven process models. Given the vagaries of requirements in COTS-based software development, developers should adopt or modify more dynamic, risk-driven spiral type process models.[4]

Because both COTS products and their market are volatile, the project team must gain an incremental understanding of the intricacies of these products (e.g., the inherent architectural assumptions and dependencies between COTS modules) as they negotiate and reconcile stakeholder needs with the characteristics and features of the technology. They should adopt an iterative, risk-driven approach, with each iterative spiral cycle focusing on resolving the most critical risks via prototyping (as much as possible) under conditions similar to those in the operational environment.[5] See Figure 1.

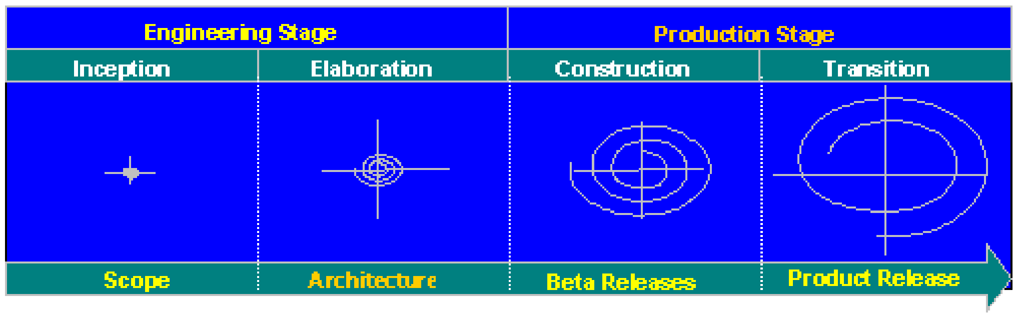


Figure 1: The RUP is an iterative, risk-driven approach.[6] ***Each iterative spiral cycle focuses on resolving the most critical risks via prototyping within iterations, and via major milestone executables between the four phases (Inception, Elaboration, Construction, Transition).***

Implementing Project X

In our case study, we will examine the implementation of Project X. An IT unit (the ITU) of a government organization (part of the Government Authority, or GA) was tasked by its management committee to source an enterprise-level, end-to-end electronic ordering and payment solution for its procurement needs. To support complete decentralized purchases for end users/customers, Project X had to include these functions:

- Front-end electronic ordering, such as buying products from a catalogue.
- Funds approval and routing workflow.
- Invoice and payment management.
- Administration and reporting capabilities.
- A B2B portal to support sourcing for products, with assistance from professional buyers, partners, and suppliers.

Evaluating COTS solutions

Prior to the start of Project X, a committee from the ITU and relevant departments of the GA organization conducted a feasibility study.

To evaluate the various COTS solutions available in the market, they used the following list, which, though not exhaustive, provides a reality check to ensure that evaluators have a good understanding of the nature of COTS products and their market.[7]

- **Upgrade issues.** The content, quality, and schedule of COTS modules and components developments, enhancements, and upgrades are driven by commercial market demands over which the individual project has no control. Upgrades for every COTS module may not be released at the same time, so awareness of integration points at which the modules interact, and the impact of module upgrades, need to be carefully assessed to guard against subtle breakages.
- **Configuration management needs.** Changes, upgrades, and releases are to be expected with COTS products. A robust configuration management system is needed to track and control configuration items such as patches, versions, product releases, and licensing information.
- **Interdependencies between COTS modules/components.** Dependencies exist between COTS modules/components. Avoid selecting COTS solutions that are tightly coupled without open/extensible interfaces. Whenever possible, aim for minimal coupling. Information on module and component dependencies will also need to be under configuration management.
- **Inherent design assumptions within COTS.** COTS solutions (their design and architecture) adopt certain underlying

assumptions, which may not match the specific project's end user needs or business processes. Evaluators should understand the gaps or divergences. These underlying assumptions differ from vendor to vendor.

- **Integration mechanisms, APIs, and means of customization.** COTS products have different integration mechanisms, open APIs, and scripting languages for customization.
- **Costs of ownership.** Total cost of ownership includes upfront acquisition costs, ongoing operation and support costs, and projected growth costs. Market changes (e.g., licensing options such as lease versus buy) are also taken into consideration over the lifespan of the COTS system.
- **End users' needs versus COTS capabilities.** Understand the end users' business processes; prioritize the non-negotiable "must-haves" and negotiable features. As Boehm and Abts advise:

> Keep requirements negotiable until the system's architecture and COTS choices stabilize. This prevents you [the developer team] from promising features and capabilities that the [COTS] system cannot support easily -- or at all. Finally, involve all key stakeholders in critical COTS decisions. These stakeholders can include users, customers, developers, testers, maintainers, operators, or others as appropriate.[8]

Be aware of the impact that COTS capabilities and limitations have on requirements; this will, in turn, have a significant impact on what actually gets delivered.

The vision document: Problem and positioning statements

During the feasibility study, team members identified and assessed potential project risks associated with COTS on the short list. At the same time, they worked on developing the project business case, measuring the total cost of ownership for each COTS solution against Project X's budget. They also initiated two activities: *Identify* and *Assess Risks*, and *Develop Business Case* (shown in Figure 2).

Based on the feasibility study, the team decided that a COTS package solution using Ariba[9] modules was the preferred choice for Project X.

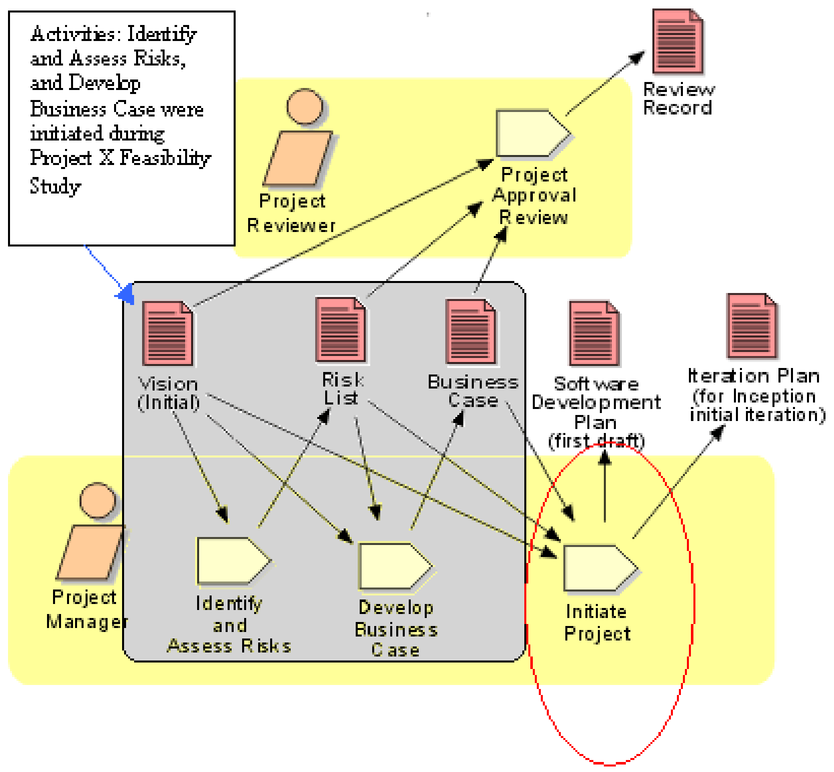

Activities: Identify and Assess Risks, and Develop Business Case were initiated during Project X Feasibility Study
Review Record
Project Reviewer
Project Approval Review
Vision (Initial)
Risk List
Business Case
Software Development Plan (first draft)
Iteration Plan (for Inception initial iteration)
Project Manager
Identify and Assess Risks
Develop Business Case
Initiate Project

Figure 2: RUP project management discipline, workflow detail: Conceive New Project

Tables 1 and 2 show the problem statement and product positioning statement matrices that the feasibility study team adapted from the RUP *Vision Document* template. Information gathered during the feasibility study contributed to the problem statement and product positioning statement of the *Vision Document*, which was subsequently created as an artifact of the requirements management discipline. Problem and product positioning statements must be clearly communicated so project participants and stakeholders have a common understanding.

Table 1: Problem statement (adapted from RUP Vision Document template)

The problem of...	*[describe the problem]* • GA organization wasting valuable resource time (an average rate of three working weeks per procurement) on paperwork and coordination to complete procurement activities. • Isolated stovepipe enterprise

	applications and backend systems with information pertaining to validated suppliers, partners, and professional buyers were scattered across different business units, making it difficult to share information and thereby control corporate expenditures (e.g., by negotiating corporate discounts).
Affects...	*[the stakeholders affected by the problem]* • the GA organization
The impact of which is...	*[what is the impact of the problem?]* • loss of valuable resource time, and delaying other business activities that depend on the products involved in the procurement for the GA organization. • not having readily available, useful information that is vital to making cost-effective procurement decisions.
A successful solution would...	*[list some key benefits of a successful solution]* • reduce paperwork and time spent on procurement activities. • decentralize procurement activities to end users. • provide seamless integration to enterprise applications and backend systems (such as those in finance, procurement, and logistics departments) so timely information is readily available for

	making cost-effective procurement decisions. • not increase the operating costs of the GA organization.

Table 2: Product position statement (adapted from RUP Vision Document template)

For...	*[target customer]* the GA organization's internal end-user community.
Who...	*[statement of the need or opportunity]* is wasting valuable resource time (an average of three working weeks per procurement) in coordination to complete procurement activities.
The System X...	*is a [product category]* is a COTS package solution (Ariba) with a combination of Web-based and client-server electronic ordering and payment capabilities, integrated with a B2B portal of validated suppliers, partners, and professional buyers to assist in sourcing of products.
That...	*[statement of key benefit; that is, the compelling reason to buy]* has extensive procurement (Ariba Buyer module) and sourcing (Ariba Sourcing and Ariba Market Place modules) functionalities and can be quickly integrated and put into operation.

Unlike...	*[primary competitive alternative]* • continuing with our stovepipe applications and heavy manual coordination approach to the business of procurement, or • building an enterprise system from scratch to solve our business problems.
Our product...	*[statement of primary differentiation]* will provide a solution that is guaranteed to solve our key business objectives of: • decentralizing purchases to end users, and shortening the number of workdays spent on procurement activities. • contributing to a unifying platform so relevant information required for procurement activities can be shared across different business units.

Based on the feasibility study results,[10] the project tender was awarded to a vendor-contractor (affiliate partner to the COTS vendor) team specializing in e-commerce and COTS (Ariba) package solution implementation. RUP best practices were recommended by the GA organization as the standard for software development. A tender compliance document was finalized in agreement between the GA organization and the vendor-contractor.

Roles and responsibilities

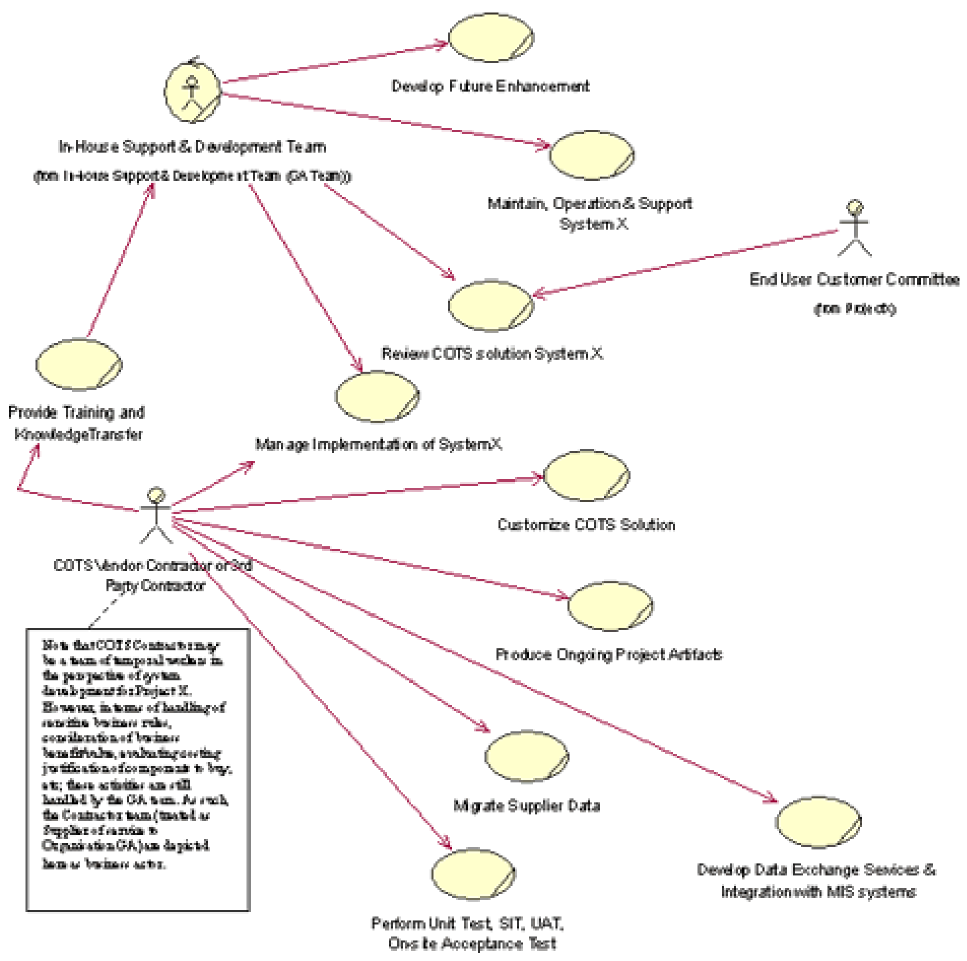
The team structure for Project X included the following parties (see Figure 2):

- **GA team** consisting of ten members, mainly from the ITU of the GA organization. This team was responsible for coordinating the project with the customer/end-user committee, reviewing the project artifacts, and providing future enhancements and maintenance for the completed system.

- **Contractor team** comprising fifteen members from the vendor/contractor. This team was responsible for the implementation of the COTS packaged solution, implementing any add-ons or customizations of the application software, and ensuring operational and seamless integration with all back-end systems.
- **Customer/End-User committee** of business representatives from the various business units and subsidiaries of the GA organization. Although end users were part of the overall organization, they were also customers of the procurement services, since there would be interdepartmental charging to finance the system's operation and support. As such, there was a clear understanding from the start of the project that requirements had to be prioritized. Any significant additional requirements had to be justified, as they would be translated to costs charged by the contractors to the GA team, and in turn to the customers/end users.

The project managers drafted a *Software Development Plan* (using RUP templates) and assigned team responsibilities (see Figure 3) and roles (Table 3). The RUP instructs teams to "Identify the project organizational units that will be responsible for each of the disciplines, workflow details, and supporting processes...."

Although Figure 3 does not depict it, for the GA Team to have a practical understanding of the COTS system, they would also need to share some of the responsibilities: Customize COTS Solution, Migrate Supplier Data, Perform Unit Test, System Integration Test (SIT), User Acceptance Test (UAT), and On-Site Acceptance Test.


Develop Future Enhancement
In-House Support & Development Team
(from In-House Support & Development Team (GA Team))
Maintain, Operation & Support System X
End User Customer Committee
(from Project)
Review COTS solution System X
Provide Training and KnowledgeTransfer
Manage Implementation of SystemX
Customize COTS Solution
COTS Vendor Contractor or 3rd Party Contractor
Produce Ongoing Project Artifacts
Migrate Supplier Data
Develop Data Exchange Services & Integration with MIS systems
Perform Unit Test, SIT, UAT, Onsite Acceptance Test

Note that COTS Contractor may be a team of temporal workers in the perspective of system development for Project X. However, in terms of handling of sensitive business rules, consideration of business benefits/value, evaluating costing justification of components to buy, etc; these activities are still handled by the GA team. As such, the Contractor team (treated as Supplier of service to Organization GA) are depicted here as business actor.

Figure 3: Overview of project teams roles and responsibilities (IBM Rational Rose® Model)[11]

Table 3: Overview of project roles (as documented within the RUP Software Development Plan [SDP])

GA Team	Contractor Team	RUP Role
GA Team: 10 members 1Senior Program Manager 1Senior Project Manager 1Architect 5 Senior Software Engineers (1 designated as DBA) 2 Software	**Contractor Team: 15 members** 1Contractor Project Manager 1Contractor Architect 3 Contractor Senior Software Engineers (1 designated as DBA) 8 Contractor Software Engineers	

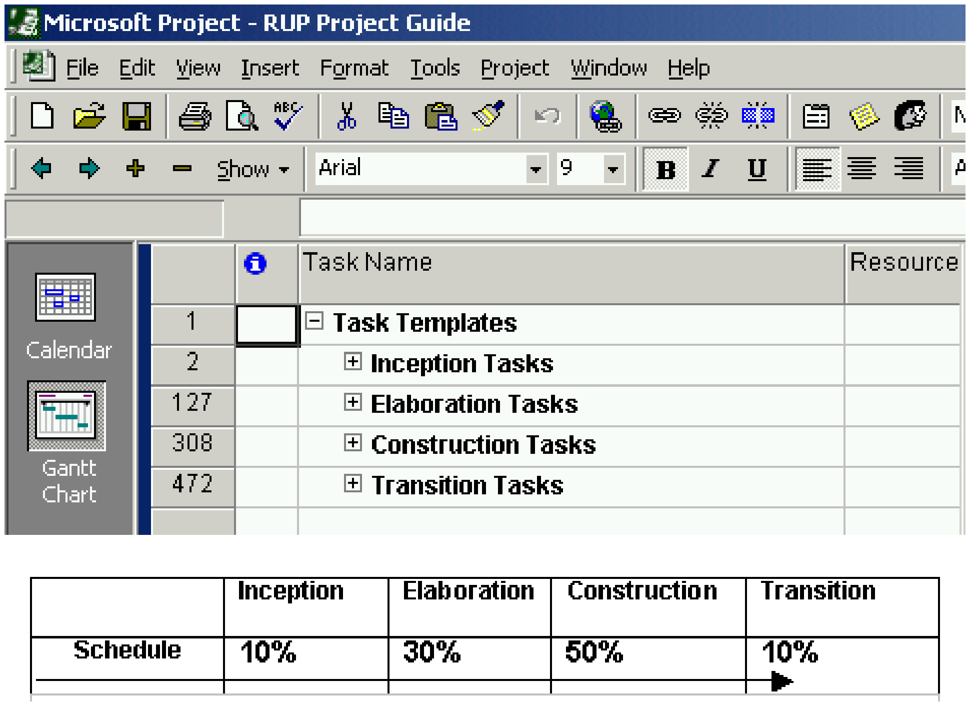
Engineers	2 Contractor Junior Software Engineers	
Senior Project Manager	Contractor Project Manager Contractor Architect	Project Manager Process Engineer Requirements Reviewer Change Control Manager
Architect	Contractor Architect	Deployment Manager Architecture Reviewer Configuration Manager
Senior Program Manager	Contractor Project Manager	Project Reviewer Requirements Reviewer
5 Senior Software Engineers	3 Contractor Senior Software Engineers	Software Architect Design Reviewer Code Reviewer Integrator Test Manager
Domain Experts from Customer/End-User Committee are included to work together with GA team and Contractor team to reach a common understanding of the business domain and processes.		Business Model Reviewer Business Process Analyst Business Designer
{GA team function more as reviewers in this section of responsibilities.}	8 Contractor Software Engineers 2 Contractor Junior Software Engineers	System Analyst[12] Requirements Specifier User Interface Designer Designer Implementer Technical Writer
1 Senior Software Engineer (DBA)	1Contractor Senior Software Engineer (DBA)	Database Designer
5 Senior Software Engineers 2 Software Engineers	8 Contractor Software Engineers 2 Contractor Junior Software Engineers	Test Analyst Test Designer Tester Technical Writer

2 Software Engineers	1Contractor Software Engineer 2 Contractor Junior Software Engineers	System Administrator Tools Specialist

Planning and scheduling

For project planning (*RUP Plan Phases and Iterations* activity), the team adapted Microsoft Project templates from RUP (see Figures 4, 5a, 5b, and 6). The project schedule, targeted at forty weeks, was divided as follows: 10 percent for the Inception phase, 30 percent for Elaboration, 50 percent for Construction, and 10 percent for Transition.[13] They also developed an iteration plan for the Inception phase (refer to Table 6 for the activities considered).

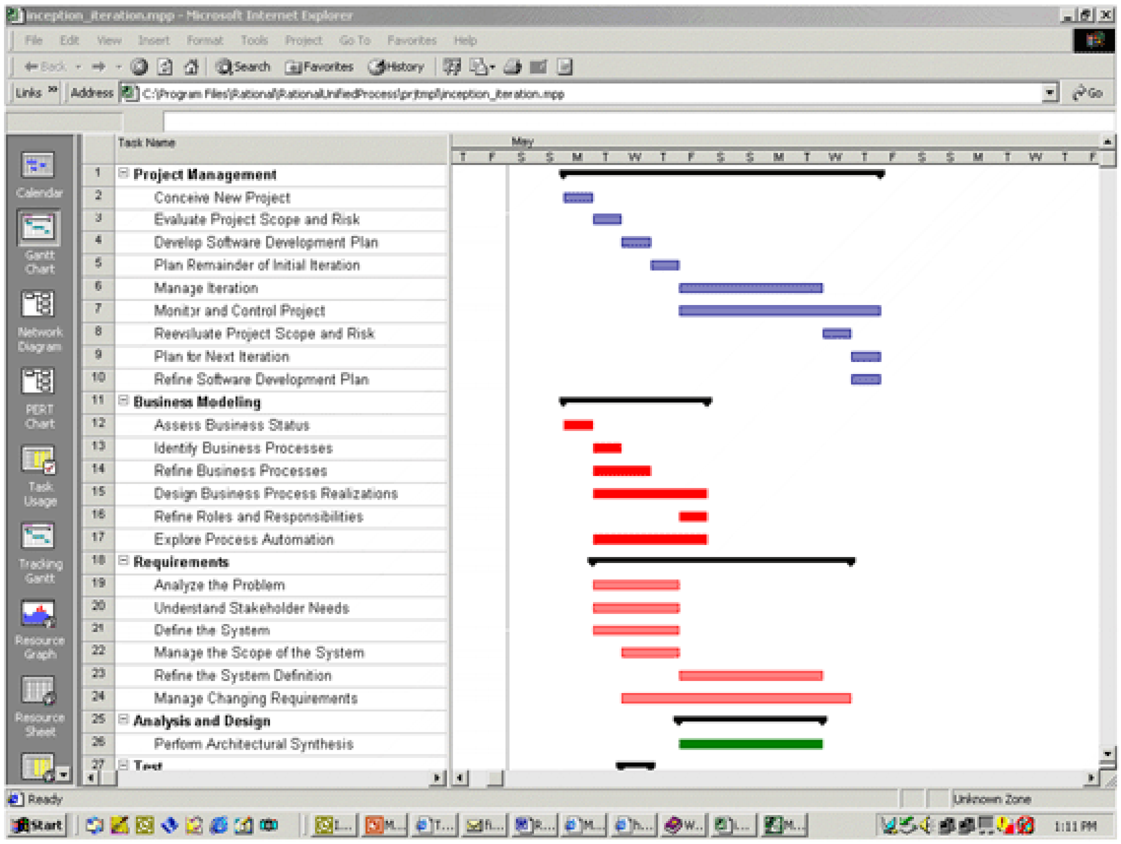
Sample templates for both Inception iteration plans and Elaboration iteration plans are also available in RUP.[14]


Microsoft Project - RUP Project Guide
File Edit View Insert Format Tools Project Window Help
Show
Arial
9
Calendar
Gantt Chart

	Task Name	Resource
1	Task Templates	
2	Inception Tasks	
127	Elaboration Tasks	
308	Construction Tasks	
472	Transition Tasks	

	Inception	Elaboration	Construction	Transition
Schedule	10%	30%	50%	10%

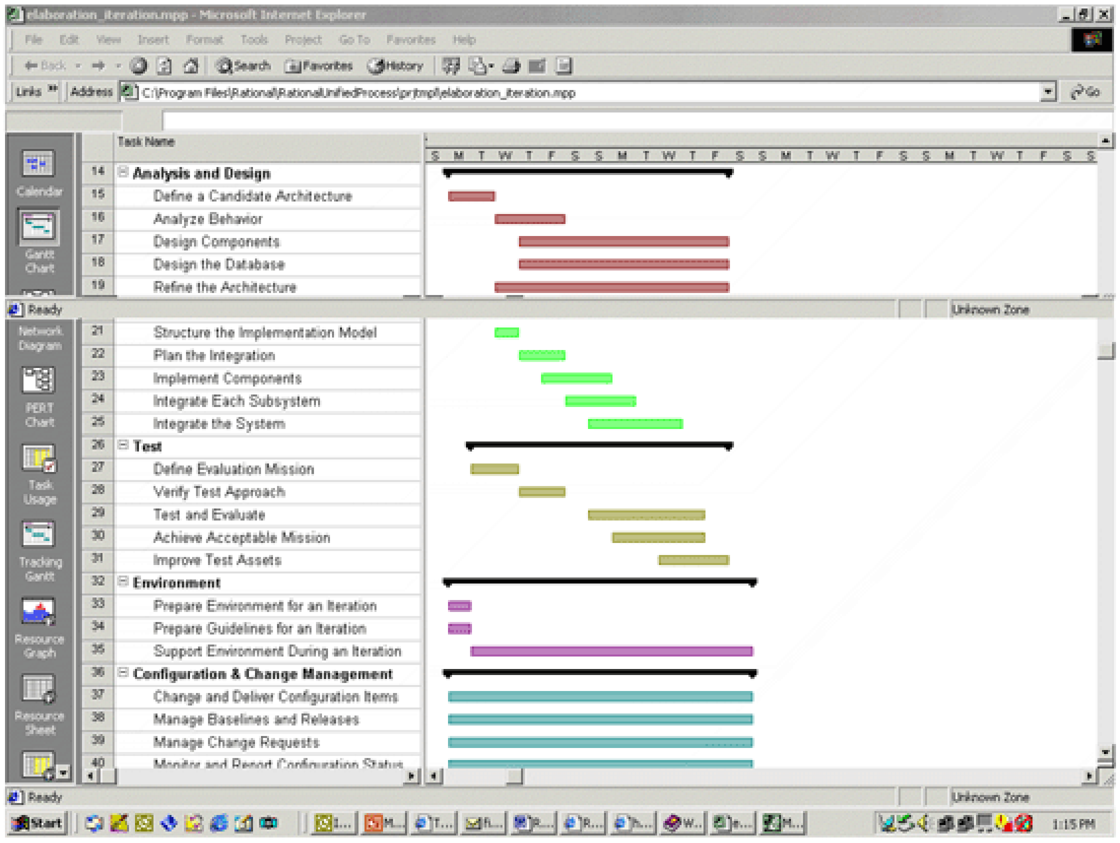
Figure 4: Division of project schedule across the four lifecycle phases


inception_iteration.mpp - Microsoft Internet Explorer
File Edit View Insert Format Tools Project Go To Favorites Help
Back Search Favorites History
Links Address
Go
Calendar
Gantt Chart
Network Diagram
PERT Chart
Task Usage
Tracking Gantt
Resource Graph
Resource Sheet
May
Ready
Unknown Zone
Start
1:11 PM

C:\Program Files\Rational\RationalUnifiedProcess\prjtmpl\inception_iteration.mpp

	Task Name
1	**Project Management**
2	Conceive New Project
3	Evaluate Project Scope and Risk
4	Develop Software Development Plan
5	Plan Remainder of Initial Iteration
6	Manage Iteration
7	Monitor and Control Project
8	Reevaluate Project Scope and Risk
9	Plan for Next Iteration
10	Refine Software Development Plan
11	**Business Modeling**
12	Assess Business Status
13	Identify Business Processes
14	Refine Business Processes
15	Design Business Process Realizations
16	Refine Roles and Responsibilities
17	Explore Process Automation
18	**Requirements**
19	Analyze the Problem
20	Understand Stakeholder Needs
21	Define the System
22	Manage the Scope of the System
23	Refine the System Definition
24	Manage Changing Requirements
25	**Analysis and Design**
26	Perform Architectural Synthesis
27	**Test**

Figure 5a: Sample Microsoft Project template for Inception iteration plan


elaboration_iteration.mpp - Microsoft Internet Explorer
File Edit View Insert Format Tools Project Go To Favorites Help
Back Search Favorites History
Links Address
Go
Calendar
Gantt Chart
Network Diagram
PERT Chart
Task Usage
Tracking Gantt
Resource Graph
Resource Sheet
Ready
Unknown Zone
Start
1:15 PM

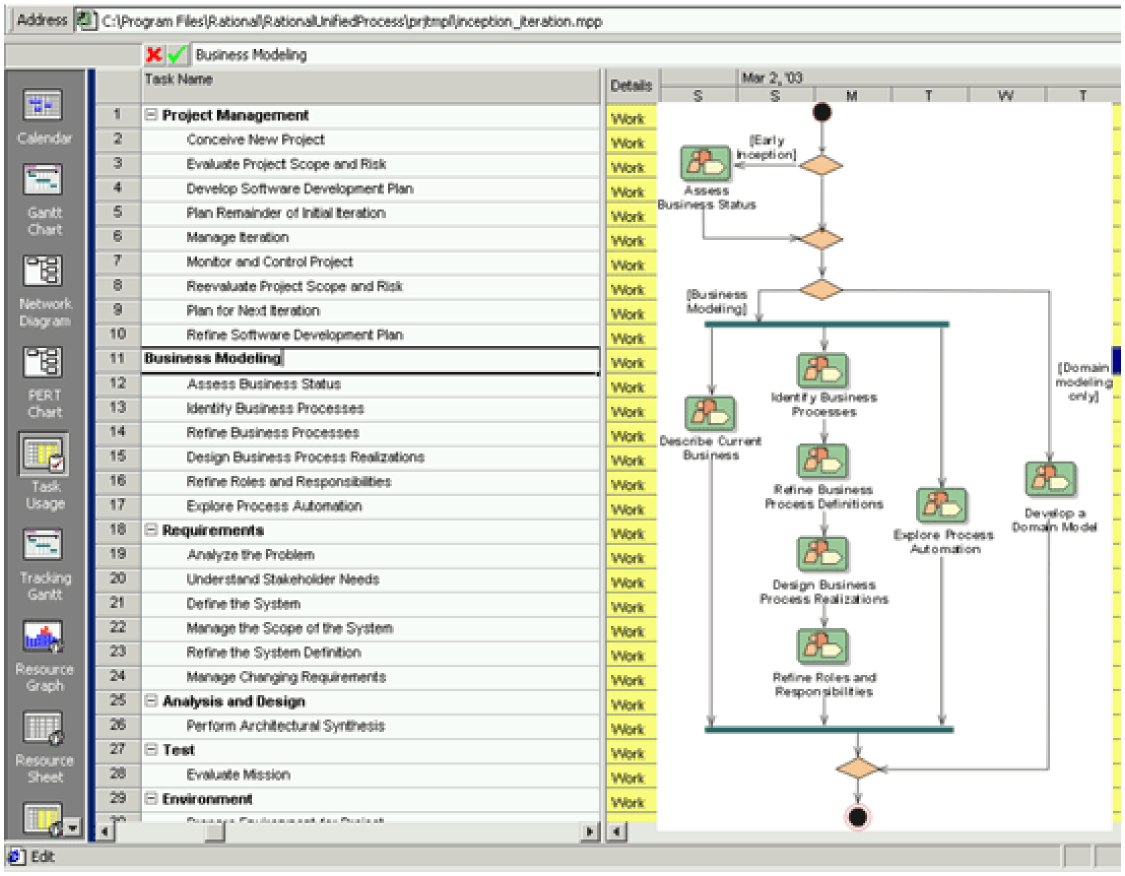
C:\Program Files\Rational\RationalUnifiedProcess\prjtmpl\elaboration_iteration.mpp

	Task Name
14	**Analysis and Design**
15	Define a Candidate Architecture
16	Analyze Behavior
17	Design Components
18	Design the Database
19	Refine the Architecture
21	Structure the Implementation Model
22	Plan the Integration
23	Implement Components
24	Integrate Each Subsystem
25	Integrate the System
26	**Test**
27	Define Evaluation Mission
28	Verify Test Approach
29	Test and Evaluate
30	Achieve Acceptable Mission
31	Improve Test Assets
32	**Environment**
33	Prepare Environment for an Iteration
34	Prepare Guidelines for an Iteration
35	Support Environment During an Iteration
36	**Configuration & Change Management**
37	Change and Deliver Configuration Items
38	Manage Baselines and Releases
39	Manage Change Requests
40	Monitor and Report Configuration Status

Figure 5b: Sample Microsoft Project template for Elaboration

iteration plan

For each task, the task templates link to RUP detail pages, which enable team members to familiarize themselves with RUP while performing their allocated tasks (see Figure 6). These details provide training for inexperienced RUP users and reinforce the knowledge of experienced users.


Address C:\Program Files\Rational\RationalUnifiedProcess\prjtmpl\inception_iteration.mpp

Business Modeling

	Task Name	Details
1	⊟ **Project Management**	Work
2	Conceive New Project	Work
3	Evaluate Project Scope and Risk	Work
4	Develop Software Development Plan	Work
5	Plan Remainder of Initial Iteration	Work
6	Manage Iteration	Work
7	Monitor and Control Project	Work
8	Reevaluate Project Scope and Risk	Work
9	Plan for Next Iteration	Work
10	Refine Software Development Plan	Work
11	**Business Modeling**	Work
12	Assess Business Status	Work
13	Identify Business Processes	Work
14	Refine Business Processes	Work
15	Design Business Process Realizations	Work
16	Refine Roles and Responsibilities	Work
17	Explore Process Automation	Work
18	⊟ **Requirements**	Work
19	Analyze the Problem	Work
20	Understand Stakeholder Needs	Work
21	Define the System	Work
22	Manage the Scope of the System	Work
23	Refine the System Definition	Work
24	Manage Changing Requirements	Work
25	⊟ **Analysis and Design**	Work
26	Perform Architectural Synthesis	Work
27	⊟ **Test**	Work
28	Evaluate Mission	Work
29	⊟ **Environment**	Work

Figure 6: Sample task template with reference links to RUP detail pages

Typically, iterations should last from two to six weeks; and, as Philippe Kruchten advises: "In general, plan to have between three and ten iterations."[15]

Applying these rules of thumb, the team came up with a preliminary plan for Project X, which had an estimated project timeline of forty weeks (excluding weekends and public holidays); the breakdown is shown in Figure 7. This preliminary project plan got the project started, and as the project progressed, the plan was incrementally refined using assessment feedback.[16]

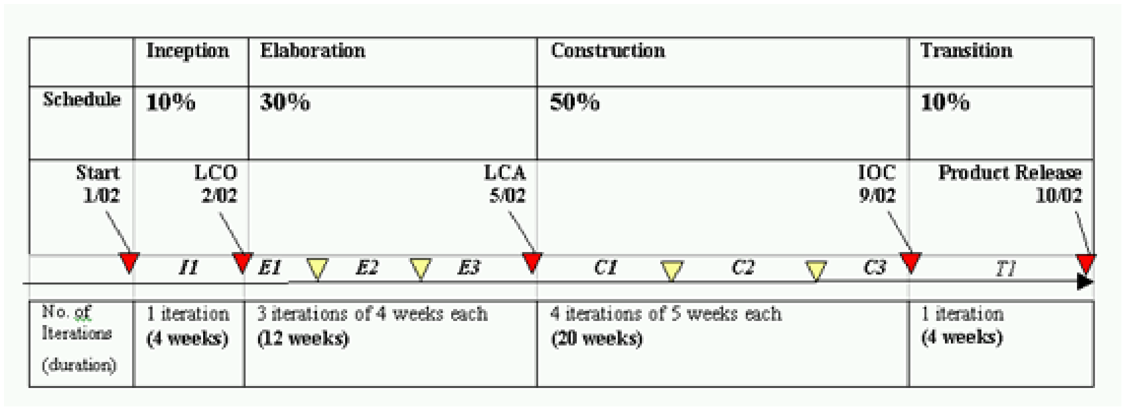


Figure 7: Preliminary schedule for Project X

Contractual payments and iterative milestones

The Rational services group deals with many project teams who intend to follow an iterative software development approach but realize, too late, that they are bound to contracts fashioned according to traditional waterfall process milestones. When contract payments are tied to delivery of particular documents (see Table 4), project teams tend to blindly use documentation as a misleading measure of project progress -- completed paperwork does not necessarily imply a completed executable. Contractors who are driven by the payment milestones stipulated in the contract will also push for document sign-off; they tend to spend disproportionate amounts of time getting each document deliverable 100 percent completed, signed off, and "frozen" before moving onto the next stage. The worst-case scenario is a project with frozen documentation that does not truly reflect the actual state of the project.

Table 4: Contractual payment milestones tied to waterfall process documents

Payment event number	Payment event	Amount to be paid (S$)	Documents tied to payments
1	Upon requirement specifications sign-off	$W	Completion of requirement specifications
2	Upon design specifications sign-off	$X	Completion of detailed design specifications
3	Upon UAT sign-off	$Y	Completion of User Acceptance Tests

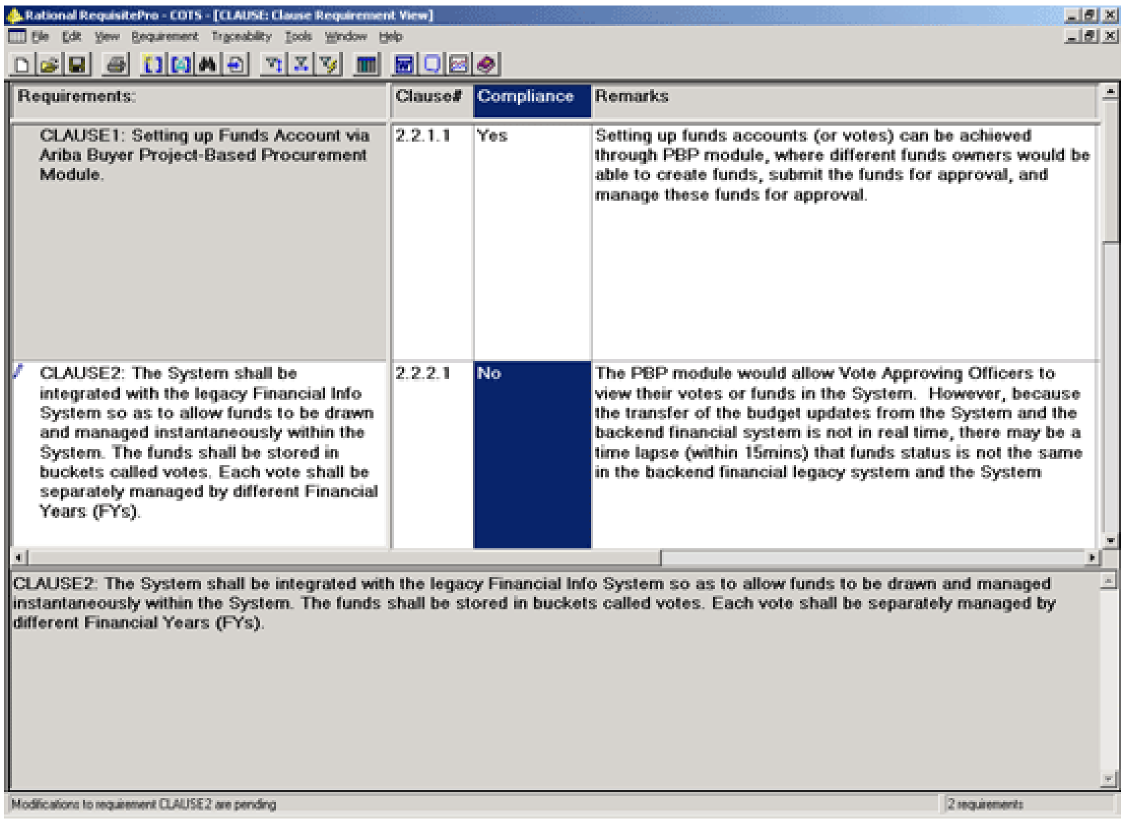
4	Upon successful system commissioning	$Z	Completion of commissioning, which includes complete delivery and deployment of the working system in the customer's environment.

To guard against this scenario, it is better to structure payment milestones in accordance with RUP milestone deliverables, which are tied to iterations (see Table 5). Each iteration ends with a working executable that can demonstrate whether the iteration criteria have been met. Project artifacts are also developed iteratively, in direct contrast to the traditional waterfall "big-bang" approach, which requires that project artifacts be 100 percent complete before moving on to the next development stage.[17] Iterations also act as natural milestones for evaluating project progress and controlling risks. Outcome artifacts associated with each phase[18] may also be noted in the contractual agreement, with the understanding that they may be developed iteratively. This is the approach the Project X team adopted.

Table 5: Contractual payment milestones tied to RUP major milestone deliverables (demonstrable executables)

Payment event number	**Payment event**	**Amount to be paid (S$)**	**Major deliverables tied to payments**
1	Inception	$A	Lifecycle Objective (LCO)
2	Elaboration	$B	Lifecycle Architecture (LCA)
3	Construction	$C	Initial Operational Capability (IOC)
4	Transition	$D	Product Release (PR)

In addition, the Project X team tracked compliance of COTS components to the contractual agreement (which could be cross-referenced back to clauses within project tender specifications) within the project repository (using IBM Rational RequisitePro® in project-defined view; see Figure 8). This enabled team members to have a common view of the contractual agreement on the COTS capabilities compliance. This view included a "Remarks" column that gave reasons or suggestions for instances of partial compliance or noncompliance.


Rational RequisitePro - COTS - [CLAUSE: Clause Requirement View]
File Edit View Requirement Traceability Tools Window Help

Requirements:	Clause#	Compliance	Remarks
CLAUSE1: Setting up Funds Account via Ariba Buyer Project-Based Procurement Module.	2.2.1.1	Yes	Setting up funds accounts (or votes) can be achieved through PBP module, where different funds owners would be able to create funds, submit the funds for approval, and manage these funds for approval.
CLAUSE2: The System shall be integrated with the legacy Financial Info System so as to allow funds to be drawn and managed instantaneously within the System. The funds shall be stored in buckets called votes. Each vote shall be separately managed by different Financial Years (FYs).	2.2.2.1	No	The PBP module would allow Vote Approving Officers to view their votes or funds in the System. However, because the transfer of the budget updates from the System and the backend financial system is not in real time, there may be a time lapse (within 15mins) that funds status is not the same in the backend financial legacy system and the System

CLAUSE2: The System shall be integrated with the legacy Financial Info System so as to allow funds to be drawn and managed instantaneously within the System. The funds shall be stored in buckets called votes. Each vote shall be separately managed by different Financial Years (FYs).

Modifications to requirement CLAUSE2 are pending

2 requirements

Figure 8: IBM Rational RequisitePro view used to track contractual tender and compliance requirements for COTS modules

The Inception iteration plan

The project team supplemented the Inception plan with details, as shown in Table 6. They carried out all the activities in agreement with both the GA and contractor teams. (For the activity *Developing the Business Case*, the contractor team contributed information on the cost of buying additional modules or add-ins). Both the GA and Contractor teams requested that the iteration plan be customized to include development case information.

Table 6: A customized Inception iteration plan. ***This plan was incrementally refined with details. Although not shown here, start and end dates and role(s) assigned for each activity are also part of the plan.***

Activities	Artifacts (and tools)	Details defined	Mentoring Activities	Review
Develop business case (refined information gathered from project feasibility study)	Business case with cost breakdown (IBM Rational RequisitePro and MS Excel)	Initial estimates	Cost estimation workshop	Formal external
• Assess target organization • Maintain business rules	• (Target organization assessment, Business rules, Business	• Business process identified/prioritized • Business process realization designed • Worker roles and	Business modeling workshop and business pattern creation	Formal external

business rules • Define business architecture • Find business actors and use cases • Find business workers and entities • Define automation requirements	rules, Business architecture document, Business use-case model) • Business object model • (IBM Rational RequisitePro, IBM Rational Rose, and IBM Rational SoDA®)	realization designed • Worker roles and responsibilities refined • Process automation defined	pattern creation	
Identify and assess risks	Risk list (with initial risk assessment), using IBM Rational RequisitePro	Risks identified	Risks identification workshop	Informal internal
Identify and assess gaps of COTS (out of box)	Gap analysis lists, using IBM Rational RequisitePro	Gaps identified	Gaps analysis workshop	Informal internal
Plan phases and iterations	Software development plan, using IBM Rational RequisitePro and Microsoft Project	Phases and milestones	Project planning workshop	Formal external
Develop requirements management plan (RMP)	RMP with requirements traceability strategy, using IBM Rational RequisitePro and IBM Rational SoDA	• Document types and requirement types • Develop requirements traceability strategy	Requirements workshop	Informal internal
• Develop vision (based on project feasibility study) • Capture a common vocabulary • Elicit stakeholder requests	• Vision document and supplementary specs • Glossary • Stakeholder requests, using IBM Rational RequisitePro	• Background and problem statement; product positioning statement and features; supplementary specs	Requirements workshop review	Formal external
Find actors and use cases	Use-case model, using IBM Rational Rose and IBM Rational SoDA	Use-case model survey	Use-case workshop review	Formal external
• Manage dependencies • Dependencies for significant requirements of COTS features and modules • Cascading dependencies between COTS modules	IBM Rational RequisitePro traceability views (and IBM Rational SoDA)	• Consistency between vision and use-case model survey established • Matrix created that maps COTS solution features to critical requirements they support	Requirements review	Informal internal

modules				
Architectural analysis	Analysis model, using IBM Rational Rose and IBM Rational SoDA	Analysis classes and mechanisms (boundary, control, and entity classes identified); enables understanding of COTS interfaces to external systems, COTS security control, and COTS persistency management.	Architecture analysis workshop	Formal internal

In general, teams refine iteration plans incrementally, based on information they gather or results they observe from earlier iterations, as shown in Figure 9.

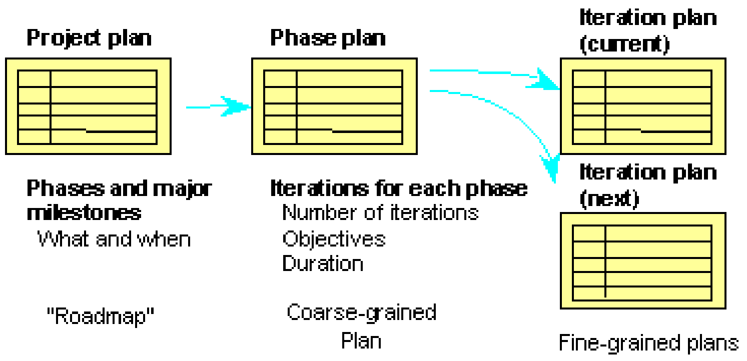


Figure 9: Incremental planning

Iteration assessment

Within an iteration, progress and risks are closely monitored to ensure that the project is aligned with the target (schedule, cost, and customer/end-user needs). Figure 10 shows the *Assess Iteration* activity as part of the *Manage Iteration* workflow detail within the RUP project management discipline.

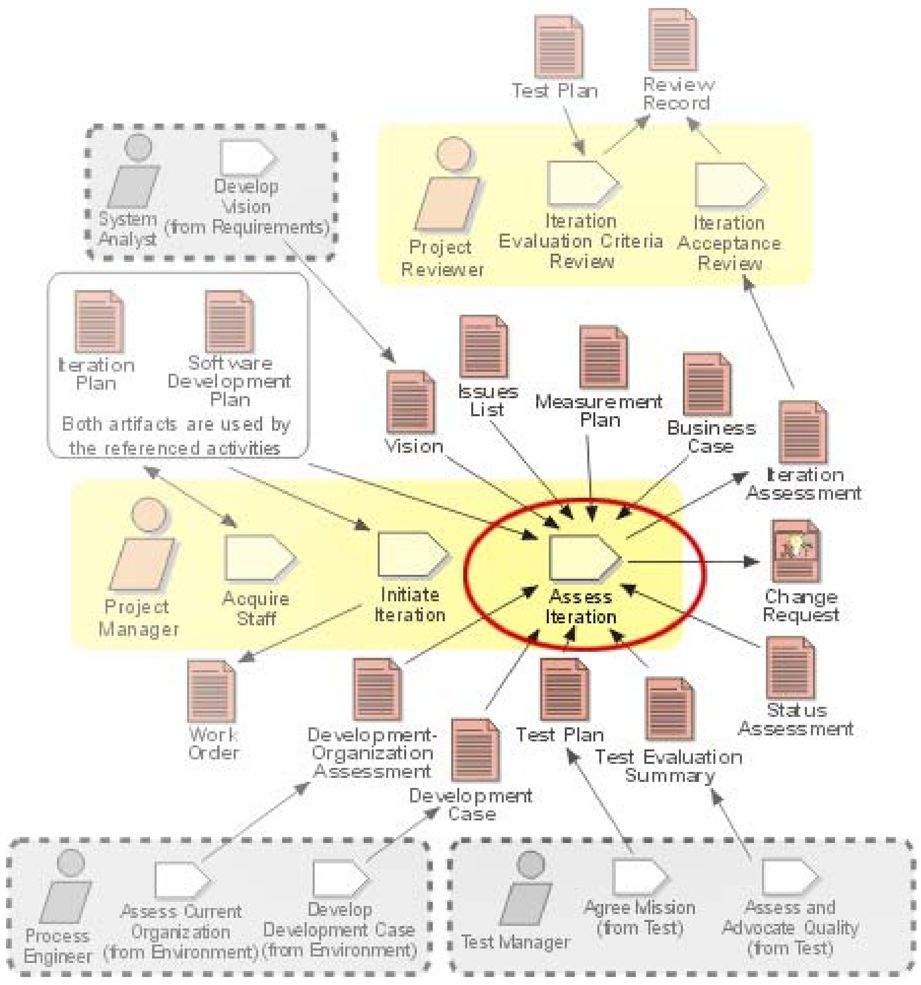

Test Plan
Review Record
System Analyst
Develop Vision (from Requirements)
Project Reviewer
Iteration Evaluation Criteria Review
Iteration Acceptance Review
Iteration Plan
Software Development Plan
Both artifacts are used by the referenced activities
Vision
Issues List
Measurement Plan
Business Case
Iteration Assessment
Project Manager
Acquire Staff
Initiate Iteration
Assess Iteration
Change Request
Work Order
Development-Organization Assessment
Development Case
Test Plan
Test Evaluation Summary
Status Assessment
Process Engineer
Assess Current Organization (from Environment)
Develop Development Case (from Environment)
Test Manager
Agree Mission (from Test)
Assess and Advocate Quality (from Test)

Figure 10: RUP project management discipline, workflow detail: Manage Iteration

Figure 11 shows an iteration assessment. Based on this project information, the project manager can respond with agility and keep the project aligned with its targets.19

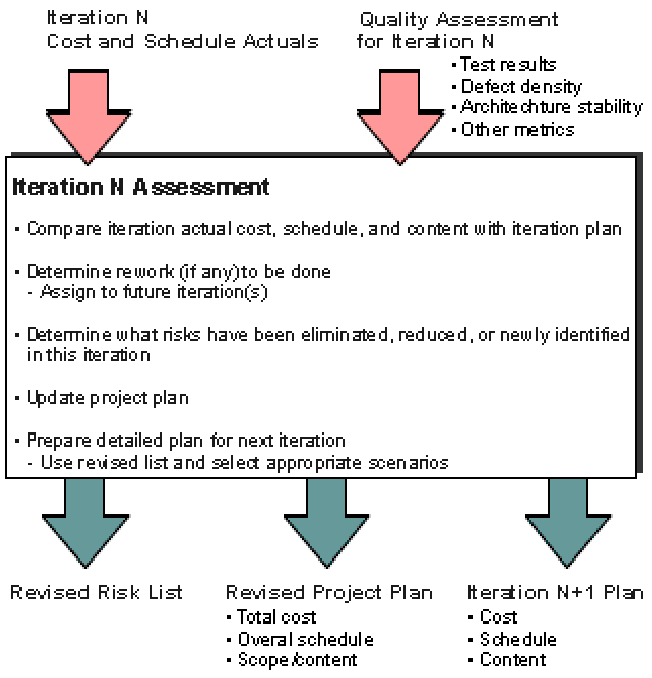

Iteration N
Cost and Schedule Actuals
Quality Assessment for Iteration N
• Test results
• Defect density
• Architechture stability
• Other metrics
Iteration N Assessment
• Compare iteration actual cost, schedule, and content with iteration plan
• Determine rework (if any) to be done
- Assign to future iteration(s)
• Determine what risks have been eliminated, reduced, or newly identified in this iteration
• Update project plan
• Prepare detailed plan for next iteration
- Use revised list and select appropriate scenarios
Revised Risk List
Revised Project Plan
• Total cost
• Overal schedule
• Scope/content
Iteration N+1 Plan
• Cost
• Schedule
• Content

Figure 11: Summary of work in a typical Iteration assessment

Here's an analogy from navigation: A good pilot plans his route ahead of a sail by looking at plots available for reference (e.g., navigation charts and historical information on wind and weather conditions along the route). But once he actually takes the helm, he must respond dynamically to changing conditions. Daily weather reports should influence his sailing strategy, but he must also be aware that weather reports may differ from actual conditions. The sailboat's actual course might not be a straight line from one point on the map to the next point. The sailboat may change directions because of changes in wind direction, weather conditions, or passenger needs. Similarly, a software project manager may shift course because of changes in technology, project conditions, or customer demands.

Process essentials

To maintain maximum agility and responsiveness, a project manager (captain) should keep the team's process as "light" as possible without losing the benefit of best practices.[20] Figure 12 shows a list of RUP process essentials.

1. Vision--Develop a vision
2. Plan--Manage to the plan
3. Risks--Mitigate risks and track related issues
4. Business case--Examine the business case
5. Architecture--Design a component architecture
6. Prototype--Incrementally build and test the product
7. Evaluation--Regularly assess results
8. Change requests--Manage and control changes
9. User support--Deploy a usable product
10. Process--Adopt a process that fits your project

Figure 12: RUP process essentials list

Those of us in the Rational services organization know that some projects tend to waive essential RUP project artifacts. We frequently hear things such as, "Since COTS solutions are plug-and-use modules, they are meant to be treated as black boxes -- they don't require any form of documentation except the user manual. In a pinch, there is always the support and maintenance contract."

Experience tells us that this is not sound reasoning. More often than not, for an enterprise COTS to be truly efficient within a particular organization, it needs customization. This is especially true for projects that span several business areas and have different "islands" of systems that need to integrate with the COTS products. Enterprise-level COTS products (such as Ariba, SAP, etc.) are not simple plug-and-play add-ins that you can use right out of the box. Customizing or extending these solutions is no easy task; without project artifacts to shed light into the COTS black boxes, future project teams responsible for later enhancements will stumble in the dark.

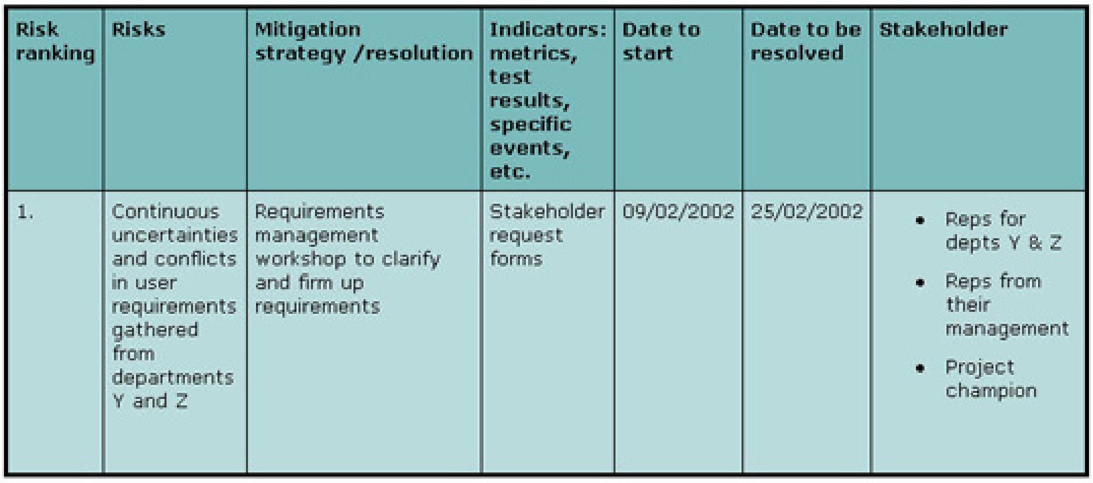
At the end of the day, the project manager needs to decide what project artifacts to include and how to customize the process to meet project needs. As an example, Table 6 shows the activities, degree of detail, and level of review Project X decided on for an Inception iteration.

Managing risks through team consensus

To manage risks, the Project X team held regular formal requirements and risk assessment workshops between iterations and included customers/end users, the contractor team, and the GA team.

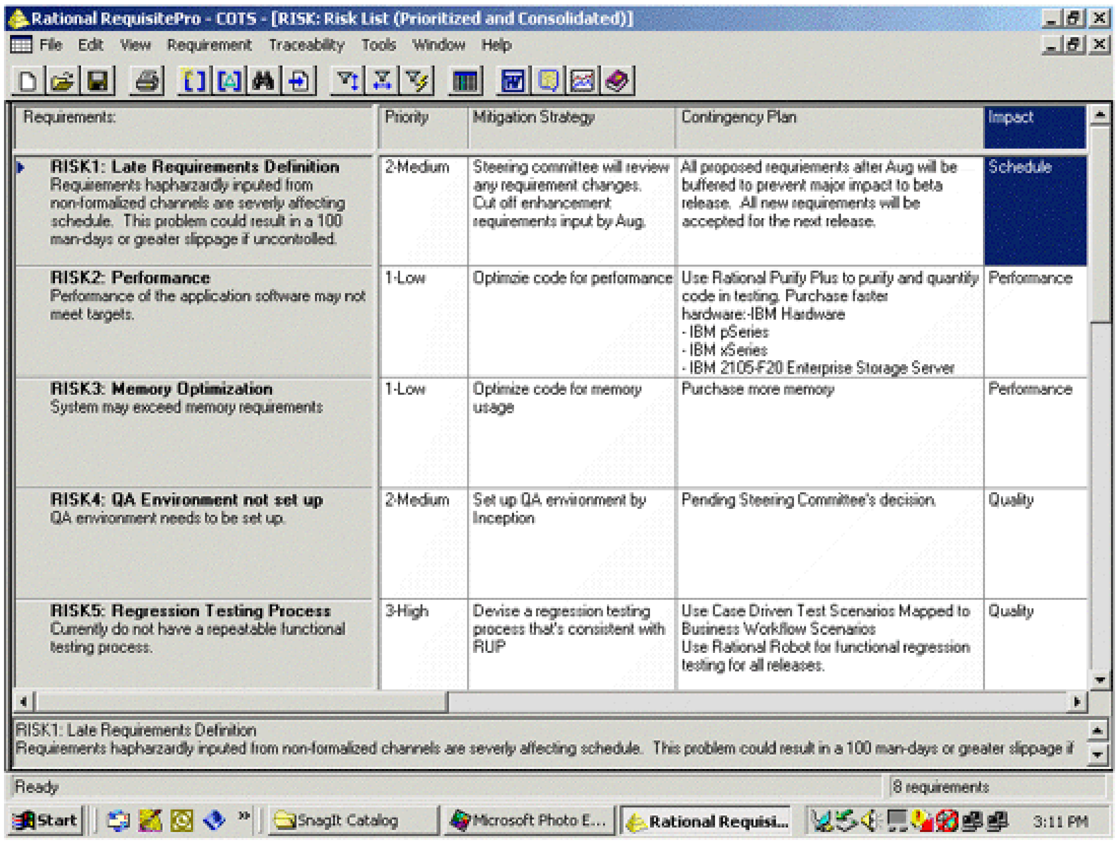
In addition the GA team conducted regular internal risk identification and assessment as part of its team meetings. Team members filled in a list of risks (see Table 7) associated with their roles and responsibilities, including potential mitigation strategies and indicators (such as test results), datelines (start and must-be-resolved-by dates), and stakeholders (people contributing to or helping to resolve the risk).

Table 7: Example risk list item. ***This submission from a team member was ranked relative to other risks, based on team consensus.***



Risk ranking	Risks	Mitigation strategy /resolution	Indicators: metrics, test results, specific events, etc.	Date to start	Date to be resolved	Stakeholder
1.	Continuous uncertainties and conflicts in user requirements gathered from departments Y and Z	Requirements management workshop to clarify and firm up requirements	Stakeholder request forms	09/02/2002	25/02/2002	• Reps for depts Y & Z • Reps from their management • Project champion

To ensure that team members were interpreting risks using a consistent approach, individual risk lists were consolidated into a project risk list. Then, the exposure for each risk was derived by team consensus, including both the GA team (for business domain and process insights, sometimes with the assistance of domain experts from the customer/end-user committee) and the contractor team (for insights into the technical aspects of the COTS solution). The Wideband Modified Delphi Method21 was used to reach a consensus estimate for risk exposure as well as for other project metrics estimates throughout the project. The consolidated risk list was kept in the project repository (using IBM Rational RequisitePro in a risk list view), so team members had a common, up-to-date view for monitoring project risks (see Figure 13).


Rational RequisitePro - COTS - [RISK: Risk List (Prioritized and Consolidated)]
File
Edit
View
Requirement
Traceability
Tools
Window
Help

Requirements:	Priority	Mitigation Strategy	Contingency Plan	Impact
RISK1: Late Requirements Definition Requirements haphazardly inputed from non-formalized channels are severly affecting schedule. This problem could result in a 100 man-days or greater slippage if uncontrolled.	2-Medium	Steering committee will review any requirement changes. Cut off enhancement requirements input by Aug.	All proposed requirements after Aug will be buffered to prevent major impact to beta release. All new requirements will be accepted for the next release.	Schedule
RISK2: Performance Performance of the application software may not meet targets.	1-Low	Optimzie code for performance	Use Rational Purify Plus to purify and quantify code in testing. Purchase faster hardware:-IBM Hardware - IBM pSeries - IBM xSeries - IBM 2105-F20 Enterprise Storage Server	Performance
RISK3: Memory Optimization System may exceed memory requirements	1-Low	Optimize code for memory usage	Purchase more memory	Performance
RISK4: QA Environment not set up QA environment needs to be set up.	2-Medium	Set up QA environment by Inception	Pending Steering Committee's decision.	Quality
RISK5: Regression Testing Process Currently do not have a repeatable functional testing process.	3-High	Devise a regression testing process that's consistent with RUP	Use Case Driven Test Scenarios Mapped to Business Workflow Scenarios Use Rational Robot for functional regression testing for all releases.	Quality

RISK1: Late Requirements Definition
Requirements haphazardly inputed from non-formalized channels are severly affecting schedule. This problem could result in a 100 man-days or greater slippage if

Ready | 8 requirements

Figure 13: Consolidated risk list captured within IBM Rational RequisitePro

RUP best practices yield success for COTS projects

RUP provides the rigor, disciplines, and software engineering best practices required for successful COTS implementation projects. With its iterative, risk-driven approach, RUP can help teams to incrementally identify, tackle, and track functionality gaps or mismatches between the COTS and customer/end-user needs. The iterative process also aligns with the evolutionary nature of COTS implementation projects.

To understand a COTS system's capabilities and gaps, project teams must engage early in hands-on prototyping in an environment that is as close as possible to the expected operational environment. Close and continuous communication should be facilitated among the in-house team responsible for COTS system maintenance, the third-party contractors, and the customer/end user. This means including workshops and reviews in each iteration plan. Iterative requirements management is also needed to match COTS capabilities to customer/end users' actual needs. Stakeholders must be able to differentiate critical "must haves" from other, more flexible requirements, so they can decide whether to accept particular inherent COTS assumptions and out-of-the-box capabilities (workflows, etc.).

Project managers can kick-start COTS implementation projects by leveraging the readily available templates and guidance within RUP. They should not waive essential process artifacts and activities under the assumption that a COTS solution is a "plug-and-play" black box. Process artifacts serve as project mechanisms to track project health and progress, and also as a means of retaining knowledge and shedding light into COTS black boxes. In addition, project managers should iteratively refine their plans and schedules in response to actual project progress.

Contractual payment for RUP projects should not be tied to traditional waterfall documentation milestones, which encourage putting off executables until the latter part of the lifecycle and circumventing the benefits of an iterative approach.

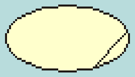
Finally project risks must be actively tackled, with a dateline for mitigation. Team consensus should be used to qualify and prioritize risks.

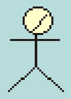
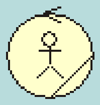
Acknowledgments

I would like to thank all my reviewers: Gary Pollice, John Smith, and Dave West for their comments and suggestions. Special thanks to Catherine Southwood for her patience and excellence in editing and tidying up my paper.

Appendix: The UML Profile for Business Modeling

Table A-1: Overview of the UML Profile for Business Modeling[22]

Stereotype	**Description**	**UML Representation**
Business use case 	A **business use case** (class) defines a set of business use-case instances, where each instance is a sequence of actions a business performs that yields an observable result of value to a particular business actor.	Use case, stereotyped as «business use case»

Business actor 	A **business actor** represents a role played in relation to the business by someone or something in the business environment.	Actor, stereotyped as «business actor».
Business worker 	A **business worker** is a class representing an abstraction of a human who acts within the system. A business worker interacts with other business workers and manipulates business entities while participating in business use-case realizations.	Class, stereotyped as «business worker»
Business use-case realization	A **business use-case realization** describes how a particular business use case is realized within the business object model, in terms of collaborating objects (instances of business workers and business entities).	Collaboration, stereotyped as «business use-case realization»
Business entity	A **business entity** is a class that is passive; that is, it does not initiate interactions on its own. A business entity object may participate in many different business use-case realizations and usually outlives any single interaction. In business modeling, business entities represent	Class, stereotyped as «business entity».

	objects that business workers access, inspect, manipulate, produce, and so on. Business entity objects provide the basis for sharing among business workers participating in different business use-case realizations.	

References

Cecilia Albert and Lisa Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview, Key Elements in Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions." (CMU/SEI-2002-TR-009 ESC-TR-2002-009), SEI, Carnegie Mellon University, July 2002.

Barry Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

Barry Boehm, and Chris Abts, "COTS Integration: Plug and Pray?" *IEEE Computer*, 32, 1 (January 1999). pp.135-138.

IBM Rational Unified Process (software product) version 2002.05.02 Rational Software Corporation. http://www.rational.com/

Philippe Kruchten, "From Waterfall to Iterative Development: A Challenging Transition for Project Managers." *The Rational Edge*, December 2000. http://www.therationaledge.com/content/dec_00/m_iterative.html

Philippe Kruchten, "Planning an Iterative Project." *The Rational Edge*, October 2002. http://www.therationaledge.com/content/oct_02/f_iterativePlanning_pk.jsp

Pan-Wei Ng, "Effective Business Modeling with UML: Describing Business Use Cases and Realizations." *The Rational Edge*, September 2002. http://field.rational.com/rattle/sept02/ng business.html

Leslee Probasco, "The Ten Essentials of RUP: The Essence of an Effective Development Process." *Rational Software Whitepaper*, September 2000.

Walker Royce. *Software Project Management-A Unified Framework*. Addison-Wesley, 1998.

John Smith, "A Comparison of RUP and XP." *Rational Software Whitepaper*, May 2001.

Notes

[1] For an introduction to RUP, see Philippe Kruchten's article, "What Is the Rational Unified Process?" in the February 2003 issue of *The Rational Edge*. Also see http://www.rational.com/products/rup/index.jsp

[2] The term *glue code* refers to code written to integrate COTS components with one another and the rest of the system.

[3] Cecilia Albert and Lisa Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview of Key Elements in Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions." (CMU/SEI-2002-TR-009 ESC-TR-2002-009), SEI, Carnegie Mellon University, July 2002.

[4] Barry Boehm and Chris Abts, "COTS Integration: Plug and Pray?" *IEEE Computer*, 32, 1 (January 1999). Pgs. 135-138.

[5] Boehm and Abts, *Op. Cit.*

[6] Figure adapted from the Rational University course"Principles of Managing Iterative Development."

[7]For more details on COTS evaluation activities, refer to Albert and Brownsword, *Op. Cit.*

[8]Boehm and Abts, *Op. Cit.*

[9]Ariba provides B2B e-commerce solutions that integrate buyers, suppliers, and B2B marketplaces, using software modules such as Ariba Buyer™, Ariba Sourcing™, and Ariba Marketplace.™

[10]The feasibility study includes practical evaluation of the different COTS products from different vendors. This may include vendors' practical proof of concept based on preliminary requirements gathered from a customer/end-user committee.

[11]Refer to the Appendix in this article: **The UML profile for business modeling.**

[12]With COTS implementations, the System Analyst role is also given the task of gap analysis, and consolidating team members' feedback on gaps, or divergence between COTS assumptions and capabilities versus the customer/end users' actual needs. This is an ongoing, iterative activity; team members gain deeper insight into the COTS system as they get hands-on experience with it.

[13]Refer to Walker Royce, *Software Project Management -- A Unified Framework*. Addison Wesley, 1998, p.148, Table 10-2, for more information on default distribution of effort and schedule by phase.

[14] RDN (www.rational.net; authorization required) also provides coarse- and fine-grained project Microsoft Project templates for RUP.

[15]From Philippe Kruchten, "From Waterfall to Iterative Development-A Challenging Transition for Project Managers." *The Rational Edge*, December 2000.

http://www.therationaledge.com/content/dec_00/m_iterative.html

[16]Depending on the degree of COTS capabilities compliance and on gaps between COTS out-of-the-box functionalities and customer/end-user needs, more time may be allocated to Construction and Transition.

[17]For more discussion on the waterfall versus iterative approach, see Philippe Kruchten, *Op. Cit*.

[18]This includes the vision document, use-case model, design model, software architecture document, etc. For details on process essentials, refer to Leslee Probasco, "The Ten Essentials of RUP: The Essence of an Effective Development Process." *Rational Software Whitepaper*, http://www.rational.com/products/whitepapers/413.jsp September 2000, and Rational Unified Process version 2002.05.02 (see http://www.rational.com/products/rup/index.jsp)

[19]For more details on steering a project see John Smith, "A Comparison of RUP and XP." *Rational Software White Paper*, May 2001.

[20]See Leslee Probasco, *Op.Cit.* and John Smith, *Op. Cit.*

[21]See Barry Boehm, *Software Engineering Economics* (Prentice Hall, 1981) and Philippe Kruchten, "Planning an Iterative Project." *The Rational Edge*, October 2002: http://www.therationaledge.com/content/oct_02/f_iterativePlanning_pk.jsp

[22]Adapted from Rational Unified Process, *Op. Cit.* and Pan-Wei Ng, "Effective Business Modeling with UML: Describing Business Use Cases and Realizations." *The Rational Edge*, September 2002.
http://field.rational.com/rattle/sept02/ng business.html

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!