

[▶ subscribe](#)[▶ contact us](#)[▶ submit an article](#)[▶ rational.com](#)[▶ issue contents](#)[▶ archives](#)[▶ mission statement](#)[▶ editorial staff](#)

## ▶ **Project planning best practices**

by [Eric Lopes Cardozo](#)

Mentor  
Empulsys

and

[DJ de Villiers](#)

Mentor  
Empulsys

*In a way, a project is like a story. Each project is different, depending on the circumstances, or the project's environment and state. Like a story, each project has objectives, and it defines actions to achieve those objectives. The results of these actions determine how the project will end. And just as the ending of a story affects readers, the execution and outcome of a project affect its stakeholders.*



*Project planning is related to all of these elements. A plan is influenced by the project's circumstances, objectives, and desired effect on its stakeholders. A plan defines a sequence of actions that leads to results. And if these actions are well planned, the project can be like a good story with a happy ending.*

## **What is project planning?**

Planning is a difficult, creative activity that involves juggling with many variables until the circumstances, objectives, and actions are in equilibrium. The result of planning is the Project Plan, which should satisfy the constraints imposed on the project. Because there are complex interdependencies among all the variables, you cannot just sit down and write a Project Plan. First, you must define an initial plan that satisfies one or two major constraints. Then you must refine this initial plan iteratively, until it satisfies all constraints. As Dwight Eisenhower put it, "The plan is nothing. The planning is everything." In other words, the final values of

the planning variables are less important than the understanding of the relationships among these variables that develops during planning.

Much of the material and many of the theories about project planning treat the activity as a recipe: Just follow the steps in the correct order, and you'll automatically produce a good plan. Nothing could be farther from the truth. Planning is an inherently iterative activity; often you must repeat steps and perform them in parallel. Typical instructions give project managers very little practical guidance regarding the instantiation of a Project Plan. This article attempts to fill that gap, by providing a set of proven best practices for planning -- with a focus on concurrent engineering<sup>1</sup> projects.

Even within that focus, we will not attempt to cover all dimensions of project planning, such as:

**Management planning.** In order to successfully execute a project, you need to plan for effective management, including project organization, communication, and quality assurance.

**Estimation.** Planning cannot begin without an understanding of the required effort and project schedule. These parameters are the result of *estimation*, which necessarily precedes planning.

**Negotiation.** Unreasonable effort and schedule estimates cannot be resolved by any amount of planning. To arrive at realistic constraints, you need to *negotiate* terms that are satisfactory to all stakeholders involved in the project.

These concerns are beyond the scope of this article. We have intentionally chosen to focus only on planning *work* because, in our experience, this is the most challenging part of project planning.

## A practical application for RUP

In this article we use an imaginary software development project to explain project planning techniques. This project uses an iterative approach, following the project lifecycle defined by IBM Rational Unified Process,<sup>®</sup> or RUP.<sup>®</sup> However, the techniques described in this article are not restricted to software development projects or RUP projects. The approach is suitable for all kinds of projects, as long as the project manager defines clear objectives, derives effort and schedule for each phase, and produces product and work breakdown structures.

We have read other articles on the subject of project planning with great interest, specifically "Planning an Iterative Project" by Philippe Kruchten<sup>2</sup> and "Planning a Project with the Rational Unified Process" by Dave West.<sup>3</sup> Both articles describe essentially the same approach to iterative project planning, although from different perspectives. It is not our intention to depart from this approach, but rather to illustrate a practical application of the planning framework these articles describe.

## Definitions

Throughout this article, we will use the terms and definitions shown in Table 1.

**Table 1: Basic planning terminology**

Term	Description
Objective	A desired condition at some future point in time. Objectives should be specific and measurable; they should <i>not</i> refer to activities or artifacts.
Effort	The labor required to deliver a product of a given size. Effort is measured in person-hours, person-days, or person-months.
Schedule	The timeframe required to deliver a software product. Schedule is measured in days, weeks, months, or years.
Estimation	The process <sup>4</sup> of determining the amount of effort and schedule required to deliver a product.
Role	A definition of the expected behavior and responsibilities of an individual or team, within the context of a project organization.
Activity	A unit of work to be performed by a role.
Resource	Things required for project execution that have limited availability, such as personnel, hardware, software, training, licenses, and so on.
Artifact	The tangible result of activities performed by the project team. An artifact can be a document, diagram, or model.
Deliverable	An output resulting from activities performed by the project team that has a value, material or otherwise, to a customer or other stakeholder.

Phase	The time between two major project milestones, during which a well-defined set of objectives is met and artifacts are completed.
Iteration	A time-boxed period during which one or more teams perform a set of related activities to produce a coherent and (partially) complete product.
Planning	The process of finding a set of variables (objectives, artifacts, resources, activities, effort, schedule, etc.) that satisfy all constraints imposed upon a project.
Project lifecycle	A sequence of defined stages over the full duration of a project.

## Simplifications

To describe the complex process of planning in a simple and understandable manner, we will use the following conventions:

- Schedule is always measured in weeks.
- Effort is always measured in person-hours.
- Effort is always rounded up or down to reflect the level of uncertainty.<sup>5</sup>
- Phases and iterations consist of whole weeks (starting on Monday and ending on Friday).

## A multi-level approach

This article consists of two main sections:

1. Project planning
2. Iteration planning

One best practice for project planning is multi-level planning, or developing plans at different levels of detail. These levels are necessary because not all stakeholders are interested in the same information. Executives and customers are interested in high-level planning information, such as the overall budget and schedule. The project manager and project team both need very detailed information in order to execute and manage the project. Furthermore, different levels of detail

are also very useful for effective project planning. The combination of high-level (top-down) and detailed (bottom-up) planning is key to iteratively refining the Project Plan.

## **Project planning**

The result of project planning is a coarse-grained plan, or roadmap, for the project that defines boundaries regarding objectives, budget, schedule, and activities. A good high-level Project Plan will remain roughly intact no matter how the iterations are planned, as long as the detailed iteration plans fit within the Project Plan boundaries. You can develop a Project Plan by performing a set of basic activities:

1. Define project objectives
2. Define phases
3. Partition project effort and schedule across phases
4. Prioritize risks
5. Define milestones
6. Define iterations
7. Partition project effort and schedule across iterations

Remember that, in general, you must perform these activities repeatedly, and not necessarily in the sequence shown here.

### **Define project objectives**

Before we can write a plan, we must exactly know what we want the project to achieve. We can derive project objectives from the project's business case: The part of the business problem that the project will solve becomes the project objectives. Project objectives describe *what* is to be done, and not *why*. The motivation for achieving these objectives should be documented in the project's business case.

Project objectives should be specific, measurable, and realistic. They should start with a verb, as follows:

- Develop a Web-based system for handling customer complaints in order to...
- Roll out the system for all companies...
- Enable a 10 percent decrease in turnaround time for...
- Achieve a 10 percent decrease in turnaround time for...

Project objectives alone are not enough to measure the success of a project. You need to define project acceptance criteria that ensure all stakeholders will be satisfied once you achieve the project objectives. These criteria will also prevent you from forgetting something important.

In fact, the project acceptance criteria will also help stakeholders determine whether the project objectives have been achieved.

Always formulate project acceptance criteria as closed (yes/no) questions:

- Has the system been delivered to the customer?
- Is the system operational?
- Have all the agreed upon deliverables been delivered?
- Have all the agreed upon deliverables passed the customer's internal quality reviews?
- Has the project been completed within the agreed upon schedule and budget?
- Can end users use the system effectively?
- Is the support department able to provide support for the system?
- Is the historical data accessible and accurate?
- Can the previous system be switched off without impacting operations?

## **Define phases**

In this activity you describe the project's lifecycle. Phases provide project stakeholders with the opportunity to make significant decisions about committing resources to the project, one step at a time. Each phase must be concluded with a formal milestone decision -- the point at which stakeholders must decide to commit resources to achieve the objectives of the next phase (not the entire project!). The last project milestone is the point at which stakeholders decide to formally accept ownership of the project's results.

You should document the project lifecycle by splitting the project into sequential phases.<sup>6</sup> Name each phase, using a name that represents the phase's focus, which you can refine by defining phase objectives.

In this article we will use the phases described by the RUP project lifecycle. This lifecycle model suits software development projects specifically but is also very useful for other complex creative projects.

Table 2 highlights the focus for each phase defined in RUP.

### **Table 2: RUP phases and objectives**

<b>Phase</b>	<b>Objectives</b>
Inception	<ul style="list-style-type: none"> <li>• Gain stakeholder agreement on scope and boundary conditions.</li> <li>• Economically motivate the project.</li> <li>• Ensure mutual understanding of critical functionality.</li> <li>• Produce overall estimates of cost and schedule.</li> <li>• Gain insight into risks that may threaten project success.</li> </ul>
Elaboration	<ul style="list-style-type: none"> <li>• Demonstrate that the system architecture will satisfy the significant constraints imposed on the product.</li> <li>• Achieve stability of the product vision.</li> <li>• Produce a realistic plan for executing the remainder of the project.</li> <li>• Eliminate any significant risks that may threaten the success of the project.</li> </ul>
Construction	<ul style="list-style-type: none"> <li>• Build and test the system in increments.</li> <li>• Ensure increasing understanding and acceptance of the system by stakeholders.</li> <li>• Minimize costs and schedule.</li> </ul>

<b>Transition</b>	<ul style="list-style-type: none"> <li>• Achieve user self-sufficiency.</li> <li>• Achieve acceptable operational quality.</li> <li>• Transfer ownership of the system to support and maintenance.</li> <li>• Evaluate and conclude project.</li> </ul>
-------------------	---

### **Partition project effort and schedule across phases**

In this activity, the project manager allocates the estimated effort and schedule to the four phases. A distinct advantage to using the RUP phases is that RUP provides heuristics for effort and schedule distribution, as shown in Table 3.<sup>7</sup> These heuristics apply not only to software development projects, but also to most complex creative engineering projects. It is important to note that the data in Table 3 serves only as a rough guide and should be tuned to the specific project. For example, if the organization will be using RUP for the first time, the project will require more budget and schedule during Inception to do training and develop understanding among team members.

**Table 3: Percentage of effort and schedule across RUP phases**

	<b>Inception</b>	<b>Elaboration</b>	<b>Construction</b>	<b>Transition</b>
<b>Effort</b>	~ 5%	20%	65%	10%
<b>Schedule</b>	10%	30%	50%	10%

To complete this activity, you will need the following input:

- Estimate of effort
- Estimate of schedule
- Project start or end date

You can allocate a percentage of the total effort to each phase, as shown in Table 4; round up or down to make partitioning at a lower level (iterations) easier. Then, you can partition the total estimated schedule

across the phases, and determine the start and end dates of each phase accordingly. It is useful to determine the required amount of effort per week (burn rate) for each phase to avoid sharp increases or decreases in staffing requirements. In Table 4, the burn rate for Inception is 150 hours per week, for Elaboration it is 230, for Construction it is 400, and for Transition it is 200. These values indicate acceptable staffing gradations.

**Table 4: Allocation of schedule and effort across RUP phases**

<b>Phase</b>	<b>Start date</b>	<b>End date</b>	<b>Schedule</b>	<b>Effort</b>
Inception	2003-01-06	2003-01-31	4 weeks (15%)	600 hours (8%)
Elaboration	2003-02-03	2003-03-14	6 weeks (21%)	1400 hours (17%)
Construction	2003-03-17	2003-06-06	12 weeks (43%)	4800 hours (60%)
Transition	2003-06-09	2003-07-18	6 weeks (21%)	1200 hours (15%)
<b>Total</b>			<b>28 weeks</b>	<b>8000 hours</b>

### **Prioritize risks**

Proactive risk elimination is an undisputed best practice of modern project management. You can achieve this by planning to eliminate the highest priority risks in specific phases or iterations.

Once you have identified all the significant risks, gauge the impact and probability of each one, and then sort the resulting list in descending order of severity, using the grid in Figure 1.

Impact	High	Priority 6	Priority 2	Priority 1
	Medium	Priority 8	Priority 4	Priority 3
	Low	Priority 9	Priority 7	Priority 5
		Low	Medium	High
		Probability		

**Figure 1: Grid for determining risk priorities**

This grid is useful because managers often underestimate the influence of probability on severity. But it should serve only as a guideline; you must take your risk sorting a step further by seeking the opinions of a few key project team members.

Once you have prioritized the risks, it should become obvious in which phase you should target each risk for elimination. Once again, the focus of the RUP phases will help in making these decisions. Although the highest priority risks should be tackled first, you can usually eliminate risks relating to the project environment -- such as those affecting project objectives or feasibility -- during the Inception phase. Technical risks, such as new development tools, multiple platforms, or multiple systems, are often eliminated during the Elaboration phase. You can usually eliminate risks that might interfere with the rapid production of quality software, such as scope creep or delayed integration and testing, during the Construction phase. Target risks relating to deployment and commissioning, such as legacy data conversion or user skills, during the Transition phase.

This list of prioritized risks is an important input for the next step: defining project milestones. Before proceeding, however, it may be necessary to refine the phase objectives based upon the risks.

## Define milestones

After you define the objectives, budget, and schedule for each phase and prioritize risks, then you can identify project milestones and describe exit criteria for each one. A milestone marks an important point in time at which you can assess a particular artifact, synchronize activities, or deliver a product. Milestones that mark the end of a phase (referred to as *major milestones*) also serve as important decision-making moments. At these milestones, stakeholders assess the results of the preceding phase and give formal approval to proceed with the next phase of the project. In

order to set clear expectations and ensure that each phase is assessed objectively, you must specify exit criteria in advance for each milestone.

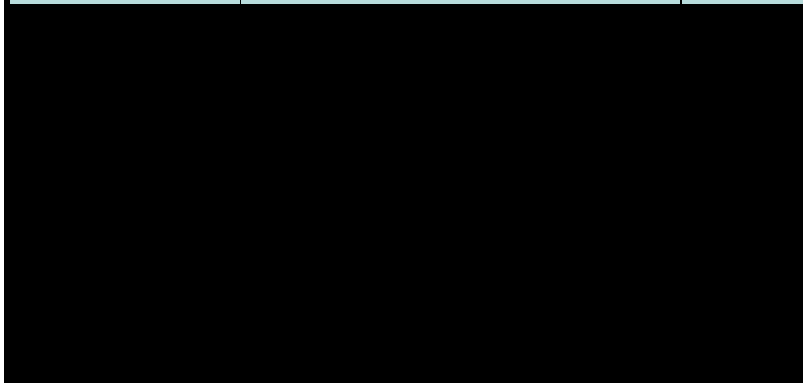
If you have done a thorough risk analysis, defining milestone exit criteria for the Inception and Elaboration phases will be easy. If you find yourself struggling to refine the default criteria described in RUP, you should revisit the *Prioritize Risks* activity.

Exit criteria for the Construction and Transition phases will be more difficult to define at this stage. Usually, the default criteria are a good place to start; you will refine these criteria sometime during Inception or Elaboration. Formulate milestone exit criteria as closed yes/no questions because you will use them to determine whether the project may proceed to the next phase. The results of this activity should look similar to Table 5.

**Table 5: Sample exit criteria for RUP milestones**

Milestone	Exit Criteria	Date
Lifecycle Objectives	<ul style="list-style-type: none"> <li data-bbox="592 819 933 1071">• Do stakeholders feel confident that the project team understands the problem to be solved?</li> <li data-bbox="592 1092 933 1270">• Do stakeholders feel confident that the major risks have been identified?</li> <li data-bbox="592 1291 933 1512">• Have the initial estimates been refined based on experiences during Inception?</li> <li data-bbox="592 1533 933 1680">• Are budget/schedule variations acceptable to the stakeholders?</li> </ul>	2003-01-31

<p><b>Lifecycle Architecture</b></p>	<ul style="list-style-type: none"> <li>• Do stakeholders feel confident that the project team is capable of delivering an appropriate solution within the parameters of the Project Plan?</li> <li>• Does the system architecture reflect both functional breadth and critical depth?</li> <li>• Have all major risks been eliminated or mitigated?</li> <li>• Are budget/schedule variations acceptable to the stakeholders?</li> </ul>	<p>2003-03-14</p>
<p><b>Beta Release</b></p>	<ul style="list-style-type: none"> <li>• Is the product sufficiently complete and of sufficient quality to start production acceptance testing?</li> <li>• Have the end-user and support organization been prepared for deployment?</li> <li>• Are budget/schedule variations acceptable to the stakeholders?</li> </ul>	<p>2003-06-06</p>



Product Release	<ul style="list-style-type: none"> <li>• Have the project objectives been realized as measured by the project acceptance criteria?</li> <li>• Are budget/schedule variations acceptable to the stakeholders?</li> </ul>	2003-07-18
-----------------	---	------------

Be sure not to include the state of a particular artifact or activity in these criteria. Simply producing an artifact or performing an activity is not in itself valuable to the project. Rather, it is what you *achieve* by producing the artifact or performing the activity that is valuable. Table 6 shows two examples of poor milestone exit criteria, as well as suggestions for improvement.

**Table 6: Examples of poor milestone criteria**

Poor Exit Criteria	Suggestion
Is the Vision complete?	Do both stakeholders and the project team agree to the problem and solution outlined in the Vision?
Have the test cases been executed?	Do stakeholders find the quality of the product acceptable?

## Define iterations

Once we have defined the phases, it is time to refine the Project Plan by defining iterations. The first step is to divide each phase into equal iterations. Iterations of the same phase do not have to be of the same length by definition, but dividing them this way provides a good starting point.

To decide on the number of iterations per phase, you can simply apply the default scheme:

- 1 Inception iteration
- 2 Elaboration iterations

- 2 Construction iterations
- 1 Transition iteration

This scheme works for most small- to medium-sized<sup>8</sup> software development projects. More information on determining the number of iterations can be found in the IBM Rational whitepaper "Planning a Project with the Rational Unified Process."<sup>9</sup> When in doubt, divide Elaboration and Construction into four-week iterations and Inception and Transition into six-week iterations.

After you determine the number of iterations, you need to define the objectives for each one. Iteration objectives are derived from the phase objectives and should therefore be very specific to the project. **Iteration objectives should eliminate risks.** You can present them as shown in Table 7.

**Table 7: Iteration objectives**

Iteration	Objectives
Inception-1	<ul style="list-style-type: none"> <li>• Achieve stakeholders' concurrence on <b>scope</b>.</li> <li>• Refine <b>estimates</b> based on actual experiences during Inception.</li> <li>• Identify and prioritize <b>risks</b>.</li> <li>• Construct a detailed <b>plan</b> for the remainder of the project.</li> </ul>
Elaboration-1	<ul style="list-style-type: none"> <li>• Develop a <b>skeleton solution</b> which covers the complex, volatile, and risky parts of the system.</li> <li>• Eliminate highest priority technical <b>risks</b>.</li> <li>• Ensure project team proficiency with <b>process and tools</b>.</li> <li>• Ensure stability of the development and test <b>environments</b>.</li> </ul>

<p>Elaboration-2</p>	<ul style="list-style-type: none"> <li>• Develop a <b>skeleton solution</b> with broad functional coverage and depth in only the risky parts.</li> <li>• Eliminate all known non-trivial <b>risks</b>.</li> <li>• Construct a detailed <b>plan</b> for the remainder of the project.</li> </ul>
<p>Construction-1</p>	<ul style="list-style-type: none"> <li>• Develop <b>80 percent of the solution</b> as rapidly as possible with reasonable quality.</li> <li>• Minimize <b>costs</b> by optimizing schedule and resources.</li> </ul>
<p>Construction-2</p>	<ul style="list-style-type: none"> <li>• Develop <b>100 percent of the solution</b>.</li> <li>• Achieve <b>acceptable quality</b> of all developed portions of the solution.</li> <li>• Prepare <b>deployment</b> to the end-user and support organization.</li> <li>• Minimize <b>costs</b> by optimizing schedule and resources.</li> </ul>
<p>Transition-1</p>	<ul style="list-style-type: none"> <li>• Improve the <b>quality</b> of the solution.</li> <li>• Ensure end-user organization <b>ability to use</b> the solution.</li> <li>• Ensure support organization <b>ability to support</b> the solution.</li> <li>• <b>Close out</b> the project.</li> </ul>

Do not be surprised if you notice that phase objectives, milestone exit criteria, and iteration objectives seem similar. These planning elements are all directly related to risks, and because we intend to actively attack project risks, this is to be expected.

### **Partition phase effort and schedule across iterations**

In order to set boundaries for producing detailed iteration plans, we need to partition the effort and schedule allocated to each phase across the iterations within that phase. For phases containing only one iteration (in RUP probably Inception or Transition) this is easy: Allocate the total effort and schedule for the phase, as in the Inception row of Table 8. For phases with more than one iteration, however, some mathematics is required.

When partitioning schedule, it is usually easiest to divide the phase into iterations of equal length. Iterations of the same phase do not have to be the same length, but this simplifies planning. When the phase cannot be partitioned into equal parts, make the first iterations longer. The team needs more time initially to understand the phase objectives and tends to work faster during succeeding iterations. We mentioned in the previous step that iteration lengths of four to six weeks are reasonable for an average project. Small projects may have iterations of two to three weeks, whereas those for large projects may be anywhere from eight to twelve weeks.

You can partition the phase effort in much the same way. Iterations of equal length within a phase usually require equal effort. Allocate more effort to longer phases and less to shorter ones, so that the total effort per week remains relatively constant within a phase. However, effort and schedule are not always allocated proportionally. For example, an early Transition iteration may be long yet require very little effort, because the product is being reviewed by stakeholders, and only a few team members are required to incorporate feedback.

**Table 8: Phase effort and schedule across iterations**

<b>Phase</b>	<b>Iteration</b>	<b>Schedule</b>	<b>Effort</b>
Inception	I1	4 weeks	600 hours
Elaboration	E1	3 weeks	700 hours
	E2	3 weeks	700 hours
Construction	C1	6 weeks	2400 hours
	C2	6 weeks	2400 hours
Transition	T1	6 weeks	1200 hours

As with phases, it is useful to determine the effort per week (burn rate) for each iteration to avoid sharp increases or decreases in staffing requirements. Using the phase effort and schedule allocations in Table 8, you can see that the Construction phase consists of two iterations, each

with a schedule of 6 weeks and an effort of 2400 hours. Division of effort by schedule results in an average of about 400 hours (or 10 people) per week during both Construction iterations.

You now have a Project Plan that you can refine at the end of each iteration. Update phase budget and schedule allocations to reflect actual progress, and refine phase and iteration objectives to reflect previously undetected risks. The number of iterations should not change unless you have significantly under- or over-estimated the project. In that case, it is quite reasonable to believe that you will require a few iterations more or less. At the start of the project, develop a detailed activity plan for the first iteration. Many project managers neglect to do this, believing that the project team already knows what needs to be done. But having a detailed plan in place within the first two weeks of the project allows you to monitor and control the project from day one. You are likely to see deviations from the plan surprisingly early -- within the first weeks of the project -- and you can adjust the overall Project Plan accordingly.

## **Iteration planning**

Near the end of each iteration, you should create a plan for the next iteration. The result of iteration planning is a fine-grained plan for a specific iteration. This plan contains the details omitted in the overall Project Plan: artifacts, activities, names, dates, and required effort.

Iteration planning consists of the following basic activities:

1. Refine iteration objectives
2. Develop product breakdown structure
3. Define evaluation criteria
4. Define iteration milestones
5. Partition effort across disciplines
6. Develop work breakdown structure
7. Staff team and assign work
8. Consolidate planning

## **Refine iteration objectives**

The overall Project Plan describes objectives for each iteration, but you may need to refine them to reflect the current state of the project. As mentioned above, you should update the Project Plan at the end of each iteration, at around the same time you plan the next iteration. The refined iteration objectives should be documented in both the Project and Iteration Plans. Table 9 provides an example of initial objectives for the first iteration of the Construction phase.

### **Table 9: Sample objectives for iteration #1, Construction phase**

Objective	Description
Develop alpha release.	Develop <b>80% of the solution</b> as rapidly as possible with reasonable quality.
Minimize costs.	Keep <b>costs as low as possible</b> by optimizing schedule and resources.

Refine iteration objectives by prioritizing risks and considering open issues. Sort risks in decreasing order of severity, and then target the highest priority risks for elimination. For more information on risk analysis, see the **Prioritize risks** section above. Table 10 shows examples of iteration objectives derived from targeting specific risks.

**Table 10: Sample iteration objectives targeting specific risks**

Objective	Description
Achieve tool proficiency.	Ensure project team is proficient with the Xyz development tool.
Approve change request procedure.	Stakeholders understand and agree to suggested change request procedure.
Understand corporate user interface standards.	Acceptance testers are familiar with the contents of the corporate user interface standards.

Because every iteration concludes with an objective assessment of the status of the project, it is reasonable to assume that a number of open issues will be carried over between iterations. These are errors or inconsistencies that have been identified during a particular iteration, but have not yet been resolved. In an iteration plan, you must consider these open issues from previous iterations and, if necessary, update the iteration objectives to reflect significant issues.

### **Develop product breakdown structure**

Objectives are met, risks are mitigated and issues are resolved by developing and refining artifacts (demonstration-based approach). Once you determine your iteration objectives, the next step is to define the artifacts you will produce to achieve these objectives. Group the artifacts

by discipline and assign an owner for each artifact. This does not mean that the owner will be the only person involved in producing the artifact, but the owner will be responsible for progress and the quality of the artifact. Table 11 shows an example of a product breakdown structure for a specific iteration.

**Table 11: Product breakdown structure (PBS) for a single iteration**

<b>Artifact Set</b>	<b>Artifact</b>	<b>Owner</b>
Management	Project Plan	P. Menhir
	Status Assessment	P. Menhir
	Iteration Plan	P. Menhir
	Iteration Assessment	P. Menhir
Requirements	Vision	I. Understand
	Use-Case Model	I. Understand
	Use Case – Confirm Payment	L. Speccit
	Use Case – Sell Stock	L. Speccit
	Use Case – Request Material	L. Speccit
Analysis and Design	Design Model	M. Archie
	Data Model	S. Tructu
	Component - Trading	D. Velepá
	Component - Ticker	O. Gramma
	Component - UserMan	O. Gramma
	Release – Alpha 1	D. Velepá
Test	Phase Test Plan	W. Kerr
	Test Cases	T. D. Senna
	Test Scripts	T. D. Senna
	Test Evaluation Summary	W. Kerr
Deployment	Bill of Materials	W. Kerr
	Deployment Plan	W. Kerr

It is optional to supply review dates for each artifact as well. We prefer to document these points in time as (minor) iteration milestones. We then use these milestones to assess the status of a number of artifacts, rather than for tracking each artifact independently.

## **Define evaluation criteria**

In order to define the desired end state of the iteration, you must formulate iteration evaluation criteria. From a quality perspective, the purpose of evaluation criteria is to provide a means of measuring the success of an iteration. Within the context of planning, defining evaluation criteria is a form of the iterative project management best practice: *Begin with the end in mind*.<sup>10</sup> The set of evaluation criteria should clearly define the end -- or success criteria -- of the iteration. Defining evaluation criteria (or at least a first cut) at this point in the planning process leads to a more internally consistent iteration plan, which will require less refining.

The phase objectives, milestone exit criteria, iteration objectives, risks, and open issues all have an influence on the evaluation criteria. Many project managers neglect to document iteration evaluation criteria, as they feel it adds little value. Our experience has shown that project teams provided with clear, measurable criteria know exactly what to do right from the start of the iteration. Project teams that have no iteration evaluation criteria to focus on tend to need a week or two to "figure out exactly what to do." As mentioned earlier, you should express iteration evaluation criteria as closed questions. Group the criteria by the appropriate discipline, as shown below.

### **Requirements**

- Is the Vision stable?
- Did stakeholders actively participate in reviewing the Vision?
- Is the requirement set complete?
- Are all inconsistencies resolved?
- Are traceability relations and requirements attributes up to date?

### **Analysis and design**

- Is the software architecture stable?
- Does the design satisfy the quality ranges defined in the design guidelines?
- Has the design of the targeted use cases been documented and reviewed?
- Have the design guidelines been adhered to?

### **Implementation**

- Has a working build been delivered to the test team?
- Has the build-deploy cycle been automated?
- Does the build satisfy the quality ranges defined in the programming guidelines?
- Is there less than 10 percent rework for the upcoming iteration?

## **Test**

- Have the requirements targeted in this iteration been incorporated?
- Have all test cases, as defined in the iteration test plan, been executed?
- Does the build reflect the expected quality?
- Have test results been analyzed and reported to the customer?

## **Deployment**

- Has a draft deployment plan been discussed with the customer?
- Has the alpha release been delivered to the customer?
- Has a bill of materials been delivered to the customer that describes the contents delivered?
- Have the reviewers been walked through the Product Acceptance Plan?

## **Configuration and change management**

- Have the iteration results been baselined?
- Are the reviewers aware of how to submit a change request?
- Have all submitted changes been addressed by the change control board?
- Have all assigned change requests been incorporated?
- Have all level 1 and 2 defects been resolved?

## **Project management**

- Has the Risk List been updated and discussed with the customer?
- Has the Project Plan been updated and delivered to the customer for approval?
- Has the Iteration Plan for the second Construction iteration been delivered to the customer for approval?
- Have the results of the iteration assessment been documented and delivered to the customer?

## Define iteration milestones

Milestones signify important points in time during the project and are used to protect the project from delays. Any important decision-making moments or decision input from external parties should be marked as a milestone. Within the project team, milestones mark hand-offs and synchronization points. A hand-off is when a set of artifacts with a particular state is handed over to another team.

The first step when defining milestones is to make a list of all significant events that will occur during the iteration. Consider important dates in the project environment, such as when a particular document will be approved by the board, or when the development environment will be delivered. Try to determine a milestone per discipline, such as the completion of the Vision or the deployment of the system to the acceptance test environment. The last iteration in a phase will always conclude with a major (phase) milestone. After determining in which week each milestone will fall, furnish the milestone with a date (and a detailed description of the exit criteria if necessary). See Table 12. In most cases, the easiest way to do this is to plan backwards, starting from the last day of the iteration.

**Table 12: Iteration milestones, dates, and owners**

Milestone	Type	Date	Owner
<b>Week 12 - March 17<sup>th</sup> - March 21<sup>st</sup></b>			
Review Use Case Sell Stock	Minor	2003-3-19	I. Understand
Review Phase Test Plan	Minor	2003-3-19	W. Kerr
Deliver Use Case Sell Stock	Minor	2003-3-20	L. Speccit
<b>Week 13 - March 24<sup>th</sup> - March 28<sup>th</sup></b>			
Review Use Case Confirm Payment	Minor	2003-3-26	I. Understand
Deliver Use Case Confirm Payment	Minor	2003-3-27	L. Speccit
Report Status		2003-3-28	P. Menhir
<b>Week 14 - March 31<sup>st</sup> - April 4<sup>th</sup></b>			
Review Deployment Plan	Minor	2003-4-2	W. Kerr
Review Use Case Request Material	Minor	2003-4-2	I. Understand
Deliver Use Case Request Material	Minor	2003-4-3	L. Speccit
Review Test Set	Minor	2003-4-4	W. Kerr
<b>Week 15 - April 7<sup>th</sup> - April 11<sup>th</sup></b>			
Report Status	Minor	2003-4-11	P. Menhir
Deliver Development Release	Minor	2003-4-11	D. Velepa
<b>Week 16 - April 14<sup>th</sup> - April 18<sup>th</sup></b>			
Review Deployment Plan	Minor	2003-4-14	W. Kerr
<b>Week 17 - April 21<sup>st</sup> - April 25<sup>th</sup></b>			
Review Test Evaluation Summary	Minor	2003-4-23	W. Kerr
Review Project Plan	Minor	2003-4-22	P. Menhir
Review Iteration Plan	Minor	2003-4-23	P. Menhir
Deliver Alpha Release	Minor	2003-4-24	D. Velepa
Iteration Assessment	Major	2003-4-25	P. Menhir

Iteration Assessment	Project	2003-1-20	11/20/03
----------------------	---------	-----------	----------

In Table 12, notice that only the Iteration Assessment milestone is a *major* milestone. All other milestones in this iteration are minor. The milestone type is related to the freedom to change the planned milestone date. The project manager can change minor milestones, but major milestones relate to important decision-making moments and are owned by the customer.

## Partition effort across disciplines

At this stage, we have determined what needs to be achieved during the iteration, the artifacts that must be produced in order to achieve these objectives, and the important dates during the iteration. We now need to start considering how much effort will be necessary to execute the iteration. We have already determined the effort and schedule for each iteration and documented it in the Project Plan. What we need to do now is partition the effort for the iteration across the disciplines. Each discipline is a group of logically related activities.

Table 13 provides an indication of how much of the total effort of an iteration within each phase should be allocated to the disciplines. Once again, we emphasize that this should serve only as a guideline and may be adjusted according to the specific project.

**Table 13: Estimate of effort by discipline for a single iteration**

Discipline	Inception	Elaboration	Construction	Transition
Business Modeling	10%	5%	-	-
Requirements	20%	20%	5%	5%
Analysis & Design	10%	20%	15%	5%
Implementation	5%	20%	30%	20%
Test	5%	10%	20%	35%
Deployment	5%	5%	10%	20%
Config. & Change Mgmt.	5%	5%	5%	5%
Project Management	20%	10%	10%	5%
Environment	20%	5%	5%	5%

The result of this activity is a top-down estimate of the effort required to perform all the activities for a specific discipline in this iteration. This information is provided to the teams for developing their own (bottom-up)

detailed activity plans. They also need estimates of their budget in order to focus only on doing only what is necessary. Then they can estimate what percentage of their total capability they can afford to devote to activities for each discipline Table 14 shows one way of doing such an estimate.

**Table 14: Percentage of total effort allocated to each discipline for a single iteration**

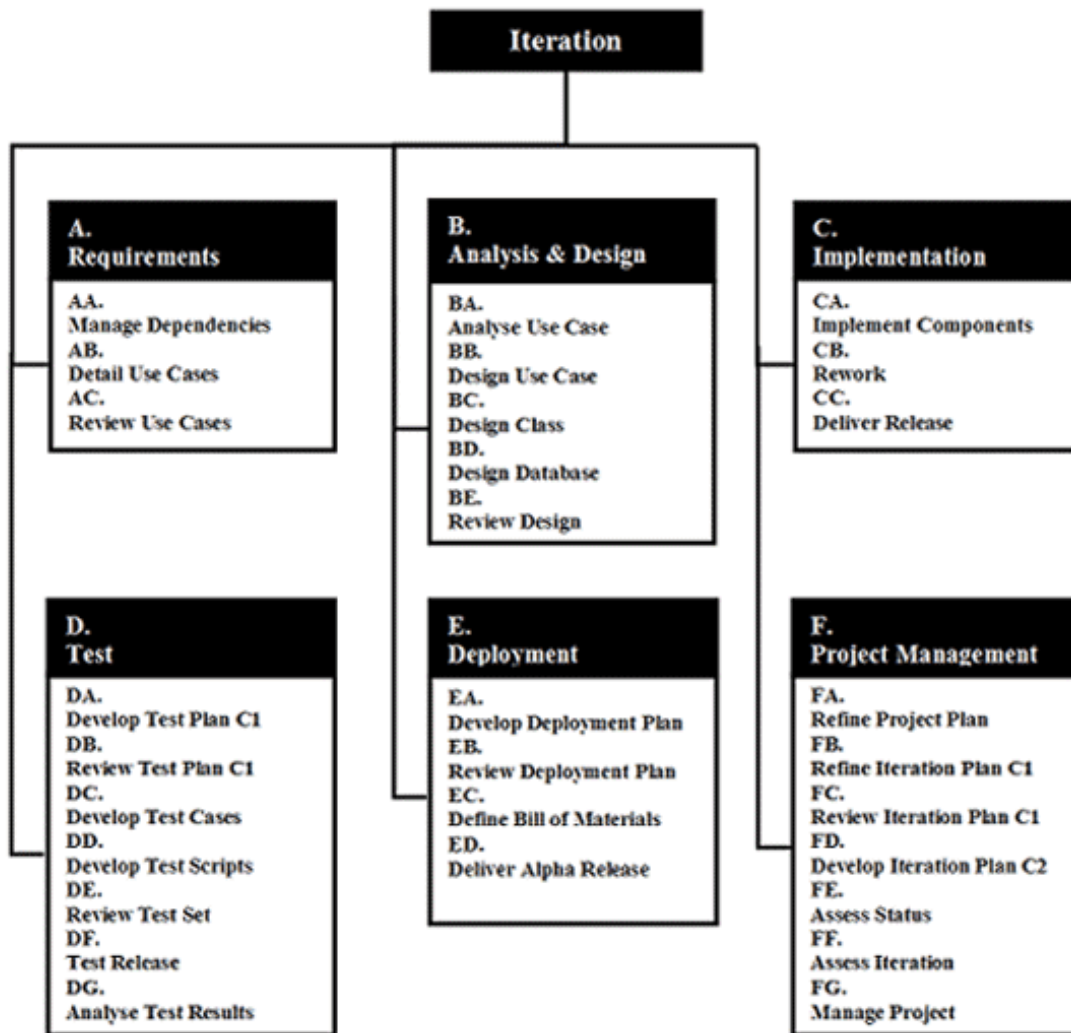
<b>Discipline</b>	<b>Effort (hours)</b>	<b>Percentage</b>
Requirements	200	9%
Analysis & Design	350	15%
Implementation	850	36%
Test	500	21%
Deployment	200	9%
Project Management	200	11%
	<b>2350</b>	

## **Develop a work breakdown structure**

The artifacts we defined above, combined with our idea of how much effort we can devote to each discipline, do not give us enough to execute the iteration. We still need to identify the activities we need to perform. It is useful to group these activities by discipline, because then we can take advantage of the partitioning we have just done. The result is a work breakdown structure (WBS), consisting of three levels: iteration, discipline, and activity. RUP itself is a very good source for defining activities, and there are distinct advantages to using a standardized WBS (not discussed here).

For each discipline, consider the iteration objectives, risks, open issues, and artifacts you need to produce, and create a list of activities based on this. For larger projects with different teams, it may make sense to let each team define its own activities. Of course, the project manager should check the results to make sure they are complete.

Do not forget to include reviews and project management activities. Some project managers prefer to define a fourth WBS level, by making each general activity product-specific. For example, you can define the *analyze use cases* activity below in terms of sub-activities: one for each use case that will actually be analyzed during the iteration. The contents of the resulting WBS should look something like Figure 2.



**Figure 2: Sample work breakdown structure (WBS)**

[Click for enlarged view](#)

Of course, it is not a requirement to document the WBS graphically. You can use a text editor or spreadsheet, as we will show in the next section.

### Partition effort across activities

Using the partitioned effort per discipline as a guide, you can now allocate the amount of effort required for each activity. Although you may feel that some knowledge of the black arts is necessary to conjure up this level of detail, in fact, this work breakdown structure is based primarily on common sense. The product breakdown (developed during the step *Define deliverables*) specifies which artifacts you must work on. The iteration evaluation criteria provide insight into what is required for these artifacts at different points in time, and the iteration milestones indicate when you should deliver them. Typically, you do not perform an entire activity within one week; most are spread out over two or more weeks.

You may notice that the totals per discipline are not exactly the same as the values you estimated earlier. This is not a problem; the top-down partitioned values simply serve as guidelines. However, you will need to resolve large discrepancies between the top-down and the bottom-up

values.

Table 15 is a detailed activity plan, which is the most important result of iteration planning. It replaces the Gantt chart traditionally used for this level of planning. Although it does not capture dependencies between activities, you can still control the schedule effectively because the iteration is of short duration and has clearly defined milestones and evaluation criteria.

**Table 15: Detailed activity plan for a single iteration**

	Week nr.	Hours per week						Totals
		12	13	14	15	16	17	
<b>Nr</b>	<b>Activity</b>							
<b>A</b>	<b>Requirements</b>							<b>168</b>
AA	Manage Dependencies	4	4	4	4	4	4	24
AB	Detail Use Cases	36	36	36				108
AC	Review Use Cases	12	12	12				36
<b>B</b>	<b>Analysis &amp; Design</b>							<b>328</b>
BA	Analyze Use Case	20	16					36
BB	Design Use Case	30	26					56
BC	Design Class	26	34	32				92
BD	Design Database	16	16	16	16	16	16	96
BE	Review	8	10	10	8	8	4	48
<b>C</b>	<b>Implementation</b>							<b>860</b>
CA	Implement Components	48	92	134	164	164	164	766
CB	Rework	38	4	6	8	8	8	82
CC	Deliver Release				4	4	4	12
<b>D</b>	<b>Test</b>							<b>528</b>
DA	Develop Test Plan C1	18						18
DB	Review Test Plan C1	2						2
DC	Develop Test Cases	48	16					64
DD	Develop Test Scripts	24	56	72	72		36	260
DE	Review Test Set	12	12	12	12			48
DF	Test Release					80		80
DG	Analyze Test Results						56	56
<b>E</b>	<b>Deployment</b>							<b>206</b>
EA	Develop Deployment Plan	16	36	32	36	36	22	178
EB	Review Deployment Plan			8		8		16
EC	Define Bill Of Materials						4	4
ED	Deliver Alpha Release						8	8
<b>F</b>	<b>Project Management</b>							<b>250</b>
FA	Refine Project Plan				4	4	4	12
FB	Refine Iteration Plan C1	8						8
FC	Review Iteration Plan C1	6						6

FB	Refine Iteration Plan C1	6						6
FC	Review Iteration Plan C1	6						6
FD	Develop Iteration Plan C2				10	10	10	30
FE	Assess Status		8		8			16
FF	Assess Iteration						12	12
FG	Manage Project	30	34	30	22	26	16	158
<b>Totals</b>		<b>412</b>	<b>412</b>	<b>404</b>	<b>368</b>	<b>368</b>	<b>368</b>	<b>2332</b>

If you check the total amount of effort per week, you will see that it matches quite satisfactorily with the 400 hours per week we estimated earlier. Of course, you would try to keep the burn rate relatively constant, so when there are significant differences, you may need to shift the milestones slightly.

### Staff team and assign work

We have almost completed our detailed iteration planning, except that we do not yet know anything about the size or constitution of the project team. We have calculated that the burn rate for this iteration is approximately 400 hours per week, or a 10-person team working 40 hours per week. Since this is a Construction iteration, some roles are not required full time. We can therefore safely assume that one to three people will be working part-time, which gives us a team size of eleven to thirteen people.

From the activity *Partition effort across disciplines*, we know how the effort is distributed over the disciplines. This information gives us a clue about the number of people required per discipline. For example, implementation takes about 30 percent of the effort, or about 720 hours in six weeks. This gives us 120 hours per week, or the equivalent of three developers. We can make similar calculation for the requirements, design, and testing roles. The result is an organization breakdown structure, shown in Table 16.

**Table 16: Organization breakdown structure (OBS)**

Quantity	Role	Team Member
1	Project Manager	P. Menhir
1	Architect	M. Archie
1	System Analyst	I. Understand
1	Requirements Specifier	L. Speccit

6	Reviewer	I. Understand M. Archie P. Menhir T.D. Senna K. Walitee D. Velepa
1	Designer	D. Velepa D. Signa
1	Integrator	D. Velepa
1	Configuration Manager	K. Walitee
5	Implementer	D. Velepa O. Gramma S. Structur D. Signa C. Oder
1	Database Designer	S. Tructur
1	Test Manager	W. Kerr
1	Deployment Manager	W. Kerr
2	Test Designer / Tester	T.D. Senna K. Walitee

During the previous step (*Partition effort across activities*), we calculated the amount of effort required per activity. The next step is to plan the work for each team member, by determining how much of this effort each person will spend.

There is a simple way of doing this when you do not have fancy planning tools. Simply create a worksheet in a spreadsheet tool to describe the overall activity planning, and then duplicate this worksheet for each team member. In each individual worksheet, you only describe the required effort per activity *for that individual person*. Then, you can sum the effort across all team members and compare this to the partitioned effort of the overall activity planning.

Last but not least, notice that steps *Partition effort across activities* and *Staff team and assign work* must be performed a number of times in order to make the product breakdown, work breakdown, and organization breakdown structures consistent. The artifacts defined in the product breakdown structure will be produced by project team members defined in the organization breakdown structure, by performing the activities defined in the work breakdown structure.

### **Consolidate planning**

The purpose of the final step is to verify consistency among objectives, deliverables, milestones, and activities. The following questions must be

answered with a firm YES!

- Are the deliverables consistent with the iteration objectives?
- Are the iteration evaluation criteria consistent with the iteration objectives?
- Are the milestones consistent with the deliverables?
- Are the activities consistent with the deliverables?
- Have all significant differences between top-down and bottom-up estimates of effort been resolved?
- Does the team constitution reflect the required amount of effort per discipline?

In this article we have presented a sequence of planning activities, but keep in mind that, in reality, many activities are actually performed simultaneously, and some are repeated until a plan evolves that satisfies the questions listed above. Once you have gained sufficient experience with the techniques outlined here, you will find yourself performing these steps as one indistinguishable blur. Until then, however, it is useful to be consciously aware of which step you are performing and to follow the guidelines described here.

## Conclusion

Applying new techniques for the first time, without experience to fall back on, may be intimidating. The measures you gain from experience are the "black magic" project managers often refer to. With the techniques described in this article, you can acquire that "black magic." By taking an educated guess and carefully monitoring the first weeks of the project, you can refine these *guesstimates* to represent project-specific indicators.

You need courage and belief to plan and manage uncertainty. Jules Verne's *belief* led to the 1865 publication of *De la Terre à la Lune*, a novel about a fantastic journey to the Moon. In 1969 Neil Armstrong had the *courage* to become the first person to touch the Moon's surface. In the upcoming decades we will be heading for Mars. With *belief* and *courage*, maybe you can plan a bold project yourself and explore new frontiers.

## References

Philippe Kruchten, "Planning an Iterative Project." *The Rational Edge*, October 2002.

Eric S. Lopes Cardozo, "The Seven Habits of Effective Iterative Development." *The Rational Edge*, June 2002.

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

Dave West, "Planning a Project with the Rational Unified Process." Rational

## Whitepaper, 2002.

---

### Notes

<sup>1</sup> These are projects in which many interrelated activities are performed concurrently, rather than sequentially (as is typical of *linear engineering projects*). Software development and research projects are examples of concurrent engineering projects.

<sup>2</sup>Kruchten, Philippe. "Planning an Iterative Project." *The Rational Edge*, October 2002.

<sup>3</sup>West, Dave. "Planning a Project with the Rational Unified Process." Rational Whitepaper, 2002.

<sup>4</sup>Many claim that estimation is an art form. With the right tools and techniques, you may not become Van Gogh, but you will sell many paintings.

<sup>5</sup>A figure of 350 or 400 conveys your uncertainty more realistically than a figure of 367,45.

<sup>6</sup>Technically, phases could be performed in parallel, or overlap somewhat, but this is usually unnecessarily difficult to manage

<sup>7</sup>From Walker Royce, *Software Project Management: A Unified Framework*. Addison Wesley, 1998.

<sup>8</sup>Between 1,000 and 20,000 person-hours with a schedule of one half year to two years.

<sup>9</sup> Dave West, *Op. Cit.*

<sup>10</sup>Lopes Cardozo, Eric S., The Seven Habits of Effective Iterative Development. *The Rational Edge*, June 2002.



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