Risk reduction with the RUP Phase Plan

by Mark Aked
Managing Consultant
Lamri Ltd.

During the Inception and Elaboration phases of a project based on IBM Rational Unified Process®, or RUP®, the project manager, ideally in collaboration with the architect, develops an artifact called the Phase Plan. A key component of the overall Software Development Plan, the Phase Plan provides a high-level, coarse-grained view of the project, showing the total number of planned iterations across the four RUP phases as well as key milestone dates for each of these iterations.

For practitioners new to RUP, a common mistake during phase planning is to decide what functionality to add during each iteration based on the sequence of the use cases. This approach overlooks the risk-driven element of RUP; it means the practitioner does not understand how to combine risk-assessment and functionality considerations in selecting content for iterations.

This article presents a disciplined approach for building the RUP Phase Plan that does consider technical risks -- those that the project team can mitigate during the Elaboration phase by implementing functionality that will manifest these risks. It also highlights ways to avoid common mistakes managers make in deriving the content of Elaboration iterations.

We will focus on the allocation of use cases for the Elaboration phase but also address some concerns related to the Construction phase. We will not cover the allocation of activities or resources to the iterations.

This article is aimed primarily at project managers and/or software architects, as these are the two roles most deeply involved in planning the project and developing the Phase Plan. It assumes some knowledge of RUP and use cases. Although we will briefly describe RUP fundamentals,
readers not familiar with the process should consult the References section below for sources.

**Definitions**

Let's begin by defining terms we will use throughout the article. These are captured in Table 1.

**Table 1: RUP and project-planning terminology**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUP or IBM Rational Unified Process</td>
<td>A comprehensive, flexible software project development framework that embodies an iterative approach and other best practices.</td>
</tr>
<tr>
<td>Use case</td>
<td>A visual description of system behavior that shows sequences of actions. A use case contains all alternate flows of events related to producing an &quot;observable result of value.&quot;</td>
</tr>
<tr>
<td>Risk</td>
<td>An ongoing or upcoming concern that has a significant probability of adversely affecting the completion of major milestones and product quality.</td>
</tr>
<tr>
<td>Issue</td>
<td>A risk that has become a reality.</td>
</tr>
<tr>
<td>Software Development Plan</td>
<td>A comprehensive, composite artifact that gathers all information required to manage the project. It encloses a number of artifacts developed during the Inception phase and is maintained throughout the project.</td>
</tr>
<tr>
<td>Risk Management Plan</td>
<td>Details how to manage the risks associated with a project. Specifies risk management tasks that will be carried out, assigns responsibilities, and identifies additional resources required for the risk management activity. For smaller projects, this plan may be embedded within the Software Development Plan.</td>
</tr>
</tbody>
</table>
### Phase Plan
A high-level, coarse-grained view of the project, developed during the Inception phase. It shows the total number of planned iterations across the four RUP phases, and key milestone dates for each of these iterations.

### Iteration Plan
A time-sequenced set of activities and tasks that includes assigned resources and task dependencies for the iteration; a fine-grained plan.

### Risk List
A list of known and open risks to the project, sorted in decreasing order of importance and associated with specific mitigation or contingency actions.

### Use-case flows
Generic term for the use-case flow of events that encompasses the normal flow of events and the alternate flow of events.

### Normal flow of events
The basic flow of events that covers what "normally" happens when the use case is performed.

### Alternate flow of events
Covers optional or exceptional behavior as well as variations of the normal behavior ("detours" from the basic flow of events).

---

**RUP in brief**

IBM Rational Unified Process, or RUP, is a process framework focusing on software development. Two tenets of the process are that it is *iterative* and *risk-driven*.

A RUP project consists of four phases: Inception, Elaboration, Construction, and Transition. Figure 1 illustrates the structure of a RUP project, showing these phases on the horizontal axis. The phases include *iterations* that proceed through each of the disciplines shown on the vertical axis. The different phases place different emphases on each of these disciplines as the project progresses through iterations. The end of an iteration (shown horizontally at the bottom of Figure 1) delivers an executable product -- either an internal or external release.
Through years of experience with software development projects, the authors of RUP identified a tendency for teams to address the easiest or best-known aspects of the project first, in order to get the project going on a sound footing, get individuals to operate as a coherent team, and build confidence. Although these are valid objectives, this approach pushes out all the unknown, risky elements toward the end of the project, so that they can easily develop into issues very close to the delivery date. Typically, this approach forces out the delivery date, adding costly, unplanned iterations to the project.

In contrast, project teams who are risk-driven don't accept this approach. Instead, they attempt to identify potential risks and tackle them early in the project lifecycle. Once they understand or even mitigate these risks, they deal with the more comfortable aspects of the project later in the lifecycle, so the project team has a far higher probability of delivering the product on time and within budget.

**Risk-driven planning**

As we saw in Table 1, the Phase Plan, a key component of the Software Development Plan, is developed during Inception. This high-level, coarse-grained view of the project shows the total number of planned iterations across the four phases, and key milestone dates for each of these iterations. Finer-grained details are captured in an Iteration Plan for each iteration.

One reason RUP is divided into four phases is to distinguish among the different focuses of activities at different times in the project lifecycle. Table 2 shows the questions we are trying to answer and the area of focus for each RUP phase.
Table 2: The focus of RUP phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key question</td>
<td>Should we build it?</td>
<td>Can we build it?</td>
<td>Are we building it?</td>
<td>Have we delivered it?</td>
</tr>
<tr>
<td>Focus</td>
<td>Scope</td>
<td>Risk</td>
<td>Functionality</td>
<td>Delivery</td>
</tr>
</tbody>
</table>

The iterations within a phase share that phase’s focus. For example, in Elaboration the primary focus is to attack risk, and the iterations within this phase are implicitly risk-focused.

Table 3 is an example Phase Plan covering Elaboration and Construction (in this example each phase consists of two iterations). Note that this tabular format is just one of a number of formats you can use to express iteration contents; others might be plain text or reports from tools such as IBM Rational RequisitePro,® for example. However, this format reinforces the point that Iteration objectives have a particular focus that is driven by the focus of the phase that the iteration is part of. We will continue to build out this table throughout this article.

Table 3: Example Phase Plan structure

<table>
<thead>
<tr>
<th>Phase</th>
<th>Elaboration</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus</td>
<td>Risk</td>
<td>Functionality</td>
</tr>
<tr>
<td>Iteration Number</td>
<td>Iteration 1</td>
<td>Iteration 2</td>
</tr>
<tr>
<td>Risks being mitigated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use-case flows to implement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milestone date</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RUP takes a quite general approach to risk-based planning. It basically guides a practitioner to identify risks, identify strategies for mitigation, include risks when prioritizing use cases, and document these risks as goals for assessing the iteration. It provides few specifics on how to actually do this. In this article, we propose a specific technique for explicitly assigning use cases to risks in order to provide documented
justification for determining the content for iterations. Although this approach may not be appropriate for every project (i.e., the project manager may be able to do this activity informally, in his or her head!), it is useful when documented justification is required, to encourage planning focused on addressing risk, or simply to provide an impetus for recovery when the planning activity has stalled.

Prerequisites

This approach requires a number of prerequisite artifacts. (In RUP, you produce these artifacts before doing the Phase Plan.)

- **Risk List.** Produced during the activity to *identify and assess risks* within the Project Management discipline; provides the current risks identified for the project.

- **Use-Case Survey.** Produced during the *find actors and use cases* activity within the requirements discipline, identifying the normal and alternate flows.

- **Development Case.** Produced during the *Develop development case* activity within the environment discipline, identifying activities to be undertaken and the particular iteration strategy for the project.

This approach also uses a particular style to specify the use cases, which is presented in Appendix A.

Phase Plan development

Allocating content to each of the iterations within the Elaboration phase requires the following steps:

- Prepare the Risk List and Use-Case Survey.

- For each iteration within the plan:
  - Identify candidate use cases for Elaboration.
  - Identify candidate use-case flows for Elaboration.
  - Assign use-case flows to the iteration.

- Complete the Phase Plan.

Prepare the Risk List and Use-Case Survey

The first step is to ensure that the primary inputs for the Phase Plan are of suitable quality and contain the information required.

As we have noted, the Risk List is created in the RUP activity *Identify and assess risks* (this activity is performed again during each iteration) to identify, analyze, and prioritize the risks, and assign an appropriate risk management strategy (avoidance, mitigation, or contingency) to each risk.
At this stage we are interested only in the technical risks and those risks we have decided can be mitigated through implementing use cases. We should calculate the risk exposure for each technical risk we extract from the Risk List, and then prioritize the risks. Those risks with the highest probability of occurring and that will have the most impact on project success should have highest priority.

A risk is anything that may adversely affect the outcome of the project, and they come in many forms, from technical to political. Although the project will be able to address some of these risks, others may well be outside the project's circle of influence. We might be able to manage some of the latter risks, but the project should focus on the former. However, keep in mind that just by working in an iterative manner, the team may either flush out or mitigate risks that they thought were outside the project's influence.

Figure 2 shows an example of technical risks managed within IBM Rational RequisitePro. The values used for the *Probability* and *Impact* should be documented in the project's Risk Management Plan. This example uses a scale of 1 to 5, where 1 indicates low probability and low impact, and 5 represents high probability and high impact. Multiplying the probability value by the impact value for each risk provides a *Risk Exposure* value we can apply for prioritizing the risks.

![Figure 2: Technical risks and risk exposure calculations](Click to enlarge)

The next step is to ensure that the use cases have been assigned a functionality priority. Again, this is an important step at this stage in the
process, as it provides us with additional criteria for deciding what to do first.

The simplest approach is to list all of the use cases and set a priority value for each one. Again, you can use a scale of 1 to 5, with 1 being low priority and 5 being high priority, or essential. Conduct this activity with the system's end users or customers to ensure that you are capturing their priorities.

**Identify candidate use cases for Elaboration**

We are now ready to produce a risk-to-use-case matrix. You can do this easily with IBM Rational RequisitePro by creating a traceability matrix; set the row requirement type to Risk and the column requirement type to Use Case (see Figure 3). This matrix could also be created on paper or in a spreadsheet application, but if you have captured the textual components of use cases in IBM Rational RequisitePro, it makes sense to maintain this information in a single repository.

Using this matrix, we can examine each risk in turn by looking across the use cases and asking, "If we implement this use case, will it transform this risk into an issue"? If the answer is "Yes," then we insert an arrow (or a trace if using IBM Rational RequisitePro) at the intersection between the risk and the use case in the matrix. If the answer is "No," we leave the intersection blank.

![Figure 3: Risk-to-use-case traceability matrix](Click to enlarge)
Having carried out the risk-to-use-case mapping for all of the technical risks, we can now remove use cases without arrows (e.g., UC4) as candidates for development during Elaboration; we can safely postpone these risks until the Construction phase.

Now we can look along the row for the first risk and identify all the intersecting use cases; implementing any of those use cases will attack that risk. If we continue this exercise for each risk, it soon becomes apparent that we can attack a number of high risks by implementing just a couple of use cases -- UC1 and UC2 -- during Elaboration.

Figure 4 shows how we can use IBM Rational RequisitePro to highlight candidate use cases for Elaboration. UC1 is the prime candidate because it enables us to address four of the five highest risks. By including UC2, we can address all five of those risks.

At this point it is tempting to allocate these two use cases to the first iteration in Elaboration, communicate to the team that they are to be developed, and start detailing the use-case specification for every flow.

However, remember that earlier we said the goal of Elaboration is to implement the smallest amount of functionality that will confirm whether the high-priority risks we identified either are -- or might become -- issues. Implementing the whole use case contradicts this principle, because many of the use-case flows will not address these risks.
To give our plan discipline and to avoid this situation, we can take the use case down to the next level of detail and carry out exactly the same process we used for identifying the use cases for Elaboration, but this time on the use-case flow of events.

**Identify candidate use-case flows for Elaboration**

Using the use cases we have identified as candidates for development during Elaboration, we can now produce a second matrix, this time focusing upon the use-case flow of events (or use-case flows, as we'll now refer to them).

If we are using IBM Rational RequisitePro, we can use the parent-child relationship feature to expand the use cases, reveal the use-case flows, and produce a risk-to-use-case flow traceability matrix (Figure 5). Then, using this matrix, we attend to each risk in turn and, looking across the use-case flows, we ask the question: "If we implement this use-case flow, will it transform this risk into an issue"? If the answer is "yes," then we insert an arrow at the intersection between the risk and the use-case flow in the matrix. If the answer is "no," then we leave the intersection blank. If, as in Figure 5, we are able to identify the mitigation of each risk with an actual step within a use-case flow, then we should place the arrow at this intersection; this may help us in selecting which use-case flows to address during Elaboration.

![Figure 5: Candidate use-case flows for Elaboration](Click to enlarge)

In the example shown in Figure 5, we now have a number of choices as to
the use-case flows from UC we may wish to implement in the first Elaboration iteration in order to address four high-priority risks.

The first step in the normal flow of events (UC1.1) addresses one of the four risks (RISK6). The second step in the normal flow of events (UC1.2) addresses four of the highest priority risks (RISK1, RISK3, RISK4, and RISK6). However, we should also consider one of the alternate flows (UC1.2.2), because it addresses the same four risks. Both the normal flow (UC1.5) and alternate flow (UC1.5.1), at the fifth step in the use-case, address RISK6, so again we have a choice.

Table 4 summarizes the candidate use-case flows for Elaboration.

### Table 4: Candidate use-case flows for Elaboration

<table>
<thead>
<tr>
<th>Candidate use-case flow 1</th>
<th>Candidate use-case flow 2</th>
<th>Candidate use-case flow 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1.1</td>
<td>UC1.1</td>
<td>UC1.1</td>
</tr>
<tr>
<td>UC1.2</td>
<td>UC1.2.2</td>
<td>UC1.2</td>
</tr>
<tr>
<td>UC1.3</td>
<td></td>
<td>UC1.3</td>
</tr>
<tr>
<td>UC1.4</td>
<td></td>
<td>UC1.4</td>
</tr>
<tr>
<td>UC1.5</td>
<td></td>
<td>UC1.5.1</td>
</tr>
</tbody>
</table>

**Assign use-case flows to the iteration**

The particular use-case flow we choose to allocate to the iteration depends upon any additional criteria that may pertain to our project. For example, early on in the project we may wish to increase the confidence of some stakeholders by showing them the normal flow of events, or determine the strategy by which we will handle exceptions. This may help us select which candidate use-case flow to implement during the first iteration of Elaboration.

Once we have selected a use-case flow, the final step of the process is to capture that decision by updating the *Risks being mitigated* and *Use-case flows to implement* sections in the Phase Plan for the first iteration, as shown in Table 5.

### Table 5: Phase Plan with first Elaboration iteration content allocated
Complete the Phase Plan

Having produced the content for the first iteration of Elaboration, we will now follow the same process (it is iterative!) for subsequent iterations in the remaining project phases. If we have addressed all of the high-priority risks, the planning can concentrate on the content for the Construction phase, in which the focus is to deliver functionality. We can consider the priorities we assigned earlier to the use cases when assigning use-case flows to initial Construction iterations. If any high-priority risks remain (or new risks have been discovered), then we can address them during the last Elaboration iteration.

Table 6 shows the completed Phase Plan for our example project.

Table 6: Completed Phase Plan
<table>
<thead>
<tr>
<th>Phase</th>
<th>Elaboration</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration number</td>
<td>Iteration 1</td>
<td>Iteration 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iteration 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iteration 4</td>
</tr>
<tr>
<td>Objective</td>
<td>Risk</td>
<td>Risk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Functionality</td>
</tr>
<tr>
<td>Risks being mitigated</td>
<td>RISK1, RISK3, RISK4, RISK6</td>
<td>RISK5</td>
</tr>
<tr>
<td>Use-case flows to implement</td>
<td>UC1.1, UC1.2, UC1.3, UC1.4, UC1.5</td>
<td>UC2.1, UC2.2, UC2.3</td>
</tr>
<tr>
<td>Milestone date</td>
<td>To be completed</td>
<td>To be completed</td>
</tr>
</tbody>
</table>

Figure 6 demonstrates how we can represent this Phase Plan in IBM Rational RequisitePro by defining an *assigned to iteration* attribute against the use-case requirement type.²
Common mistakes

Practitioners adopting RUP for the first time who use this risk-driven approach to planning are often concerned about the amount of content in the Elaboration iterations. They find it disconcerting that only one or two use-case flows are addressed early in the project and fear that progress will be too slow.

Attempting iterative development for the first time does require a different approach: Contrary to intuition, constraining the content of these iterations means that we can rapidly move through all of the disciplines and deliver the iteration in a short period of time. This approach is essential for helping the development team to:

- Adapt to and accept this new iterative style of developing software.
- Gain buy-in to the new approach through a successful, early delivery of a tangible product.
- Obtain essential feedback for the process engineer that enables him or her to assess the Development Case and add or remove activities for the next iteration.
- Create a natural "heartbeat" for the project, so that a working product is regularly delivered, and assessment of the product and process (i.e., progress) become commonplace.
It is important to educate stakeholders about the objectives of the phases, especially Elaboration, in which the aim is to attack risk and not necessarily to provide functionality. Progress during Elaboration should therefore be measured by the number of risks addressed rather than by the number of use-case flows implemented. Although the latter metric is essential, it is used to measure progress during Construction, when the focus is on implementing functionality.

If, after producing the content for an Elaboration iteration, the remaining development is insufficient to keep everyone busy, we recommend that you introduce additional use-case flows addressing the same risks for that particular Iteration. This "belt and braces" approach absolutely ensures that a risk has been addressed by testing it in more than one place.

Keep in mind that the example presented in this article is a simplification of a real-world scenario; in some cases, implementing a particular use-case flow to address many risks may be more expensive at this early stage than selecting a number of flows from separate use cases to address particular risks. If this happens, and if you can implement the separate use-case flows within the time allocated for the Elaboration iteration, then go with it. You should consider these additional factors as you plan the iteration content, but keep the focus on addressing particular risks.

If you find that you must (e.g., for political reasons) add use-case flows to an Elaboration iteration that do not address high-priority risks, then simply increase the scope of the risks you address and use this as the vehicle for introducing additional use-case flows. Don't be tempted to just add another use-case flow to provide additional functionality. During Elaboration, it is important to maintain the project team's focus on addressing risks.

**Next steps**

After completing all the steps we have described, the next steps are to allocate resources to the iterations, draw up a schedule, and then fill in the milestone date section of the Phase Plan.

**Summary**

This article outlines an approach for identifying and planning the content of iterations during the Elaboration phase of a RUP project. To ensure that your planning is risk-driven during this crucial phase, observe the following practices:

- Keep in mind that the objective of Elaboration iterations is to address and attempt to drive out technical risks.

- Use the technical risks to identify the use cases that will address these risks, and then identify the flows for these use cases that address the technical risks.
Don't waste effort implementing the whole use case: Implement as little as possible to mitigate the risk, and then move on.

Don't be too ambitious: Keep the content of an iteration at a manageable size.

Maintain the team's focus on completing the current iteration's goal.

Use requirements management tools such as IBM Rational RequisitePro to capture planning decisions (e.g., "We are implementing this use case to address these particular risks") and to capture the allocation of use cases to particular iterations.

**Resources**

**Use Cases**

Various books are devoted to the topic of writing use cases. Two good ones are:

- *Writing Effective Use Cases* by Alistair Cockburn (Addison-Wesley, 2000).
- *Use Case Modeling* by Kurt Bittner and Ian Spence (Addison-Wesley, 2002).

**RUP**

For more information on IBM Rational Unified Process, see


You'll find additional information on RUP in IBM Rational whitepapers (www.rational.com/products/whitepapers/index_all.jsp) and archived Webinars (www.rational.com/events/webinars/index.jsp) as well as the archives of The Rational Edge (www.therationaledge.com).

**Notes**

1 We refer to this as the "while I'm in there" syndrome, as it is easy to think, "While I have..."
the code on the screen, I might as well do a little bit extra." Team members should avoid this tendency, as it not only distracts them from the goal of the iteration, but also it can introduce errors that prolong the iteration and the Elaboration phase.

Note that we will always implement a flow in its entirety. A fragment of a flow is not testable in isolation, so by implementing it we cannot objectively assure that we have indeed mitigated the target risks.

The assigned to iteration attribute is a multiple value list type with the values of 1, 2, 3, and so forth, corresponding to the iteration number. A multiple list is used to enable the iteration numbers to be represented at the use case level.

Appendix A: An effective use-case specification style

You can use various styles for documenting a use case, and we are not here to recommend a particular style. However, to be able to allocate a certain use-case flow of events to a risk, you should use a style that enables you to identify each of these flows.

The style used in this article follows a simple outline numbering scheme that exploits the parent-child relationship within IBM Rational RequisitePro by allocating a unique identifier to each relevant component of the use-case specification -- for example, the use-case name, normal (or basic) flow of events, and alternate flow of events.

As Figure A-1 shows, at the top level is the use-case name, which is X. Each step in the basic flow is then named as a subset of X -- X.1 is step 1; X.2 is step 2, and so forth. Each alternate flow is then numbered at the next level as a child of the basic flow step from which it branches.

X <Use Case name>
X.1 <1st step of basic flow of events>
X.2 <2nd step of the basic flow of events>
...
X.n <nth step of the basic flow of events>
X.1.1 <1st alternate flow from 1st step of the basic flow of events>
X.1.2 <2nd alternate flow from 1st step of the basic flow of events>
X.n.1 <1st alternate flow from nth step of the basic flow of events>

Figure A-1: Numbering scheme for use-case flows

Figure A-2 shows how the numbering scheme works for a use case called Borrow Item.
Figure A-2: Example of a numbering scheme for a specific use case

An advantage of this style is that the numbering allows the reader to immediately identify the particular component of the use-case specification. For example, 1.2.1 is an alternate flow from step 1.2 of the normal flow in use case 1.

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