

Managing Iterative Software Development

Managing Iterative Software Development.....	1
Introduction.....	1
Starting a RUP project	1
Step 1 – Define the Development Case.....	1
Step 2 – Produce outline plans	2
Step 3 – Plan Artefacts.....	3
Step 4 – Assign Roles	4

This is the third in a series of three articles about modern software development techniques and management. This article focuses on the day-to-day management of a software progress and assumes that the Rational Unified Process (RUP) is the chosen development process.

Introduction

RUP is an iterative software development life cycle framework. This article is aimed at the Project Manager who is just about to start his/her first project using this framework. In reality, this would be a brave thing to do without the aid of an experienced mentor to act as a safety net (and source of excuses!).

RUP itself contains all the details one would need to begin and run a RUP project. However, it's only when you know your way around that it becomes easier to find this information. We also assume that the Project Manager, as well as all other project team members, have had some basis training in RUP. Therefore, the purpose of this article is to shine some light on the basic activities that are required to start and maintain a RUP project. In other words, it's a kind of road map showing only trunk roads – for a street map, we'd use RUP itself.

Starting a RUP project

We are going to document this in a series of basis steps. Don't forget, all the details for each step may be found in the process itself and you are strongly advised to look it up for yourself.

Step 1 – Define the Development Case

If the whole of the Rational Unified Process were to be used on a project, it would overwhelm the project in administration and unnecessary paperwork. The purpose of the Development Case is to provide stakeholders and team members with the information concerning how RUP will be abbreviated in order to suit the specific needs of the project.

In practical terms, this means that it:

- Provides an outline plan of the project
- States the purpose of each project phase
- States which Disciplines, if any, will be excluded
- Which artefact will be produced and
- When each artefact is to be updated

The next two steps include activities that result in the information contained in the Development case

Step 2 – Produce outline plans

Producing detailed, meaningful plans at the very beginning of a project is extremely difficult – some would say futile. This is because planning has to be done on the basis of experience. In most software projects, unless the team have already done a very similar project, our estimates of effort, time and cost are likely to be based largely on educated guesswork.

Producing estimates for software projects is outside the scope of this article. However, one wit¹ quipped that “Predictions are hard, especially about the future” whilst it is also said that “There are two types of estimate: Lucy or Lousy”.

In a RUP project, we typically produce very general plans at during the Inception phase and then we update them at the end of the Elaboration phase when we understand a lot more about requirements and the software architecture. However, at this stage we might well be between 20-30% into the project.

Here’s an example of an outline project plan produced during Inception:

Phase	Start – Finish	Objective
Inception	29/6/01 – 4/7/01	Identify use cases, assign risks to use cases, plan iterations and content, propose candidate architecture
Elaboration	4/7/01 – 11/7/01	Reduce risks by 75%, demonstrate architecture
Construction	11/7/01 – 20/7/01	Build all remaining use cases, provide evidence of test results
Transition	20/7/01 – 27/7/01	Deploy application and complete user acceptance testing

Table 1 – Phase Plan

To help us a little more, RUP predicts that each phase will consume the following proportions of the project:

	Inception	Elaboration	Construction	Transition
Effort	~5 %	20 %	65 %	10%
Schedule	10 %	30 %	50 %	10 %

Table 2 – Phase Effort and Schedule

¹ Yogi Berra

Thus, it is possible to retrospectively record the fact that Inception took 10 weeks and hence predict that Elaboration, Construction and Transition might be expected to consume 30, 50 and 10 weeks respectively.

Step 3 – Plan Artefacts

We know that RUP is intended to be usable on almost any size of project. This means that for small projects, the number of artefacts that we *could* produce would be too much for the project. Part of the function of the Development Case is to customise RUP by reducing this burden.

Here's an example of a grid telling the team which artefacts shall be produced and when they will be subsequently updated. This one is taken from the Requirement Discipline:

Artefacts	When to create/update artefact				Review Details	Tools Used	Templates/ Examples
	Incep	Elab	Const	Trans			
Actor							
Boundary Class							
Glossary	X	X	X		None	Word	
Requirements Attributes	X	X	X		Formal- Internal	Rose	
Requirements Management Plan							
Stakeholder Requests							
Software Requirements Specification							
Supplementary Specification	X	X			Formal - External	Word	
Use Case							
Use-Case Model	X	X	X		Formal - External	Rose	
Use-Case Package							
Use-Case Storyboard							
User-Interface Prototype							
Vision	X				Formal - External	Word	

Table 3 – Artefact Matrix

If the artefact doesn't have an X next to it, it won't be produced. Using this approach, we reduce the total number of artefacts from several dozen to only a few important ones.

In the first instance, arriving at the most appropriate list of artefacts is largely guesswork, unless someone with some experience can be found. However, after the first couple of iterations, the iteration review process can be used to verify whether the initial decision was good or bad. Naturally, if we are producing documents that have no value, we stop. On the other hand, if communication problems are found, we might need to produce more. If in doubt, choose fewer rather than more artefacts.

Another point worth noting is that many RUP artefacts are summary artefacts. Hence, sometimes we might wish to document at the summary level – other times, the detail documents are more appropriate.

Step 4 – Assign Roles

One of the first steps in a RUP project is to assign roles. This subsequently defines every activity that project team members will be expected to carry out, when it should be done, what results, and how it may be verified. Interestingly, the Standish Group² reckon that most successful projects have no more than 6 members. With that in mind, here's a suggestion as to how you might want to assign roles to a typical development team:

Job Title	Rational Unified Process Role	Name of Project Member(s)
Project Manager	Project Manager Process Engineer Deployment Manager Requirements Reviewer Architecture Reviewer Change Control Manager	John
Stakeholders	Project Reviewer Stakeholder Requirements Reviewer	Paul, Sue, Jim
Senior Developer	Code Reviewer Software Architect Design Reviewer Configuration Manager	George
Developer(s)	Implementer User Interface Designer Integrator	Bert (UI), Sarah, George
Requirements, Analysis and Design	Requirements Specifier System Analyst Designer	David (Req's), Sarah

² http://www.standishgroup.com/sample_research/chaos1998.pdf

Tester	Test Designer Tester	David
--------	-------------------------	-------

Table 4 – RUP roles and responsibilities

The RUP roles shown above represent only a fraction of the total roles defined in RUP. However, on a small project they are probably sufficient.

You will notice that some roles are done by more than one person and some people have more than one role. This is typical of most RUP projects, indeed almost unavoidable if we are to believe the Standish Groups finding that successful project require small teams. This means that at various times, individual “wear different hats” and carry out distinctly different activities. The important point to remember is that some roles are probably best not assigned to the same person e.g. Implementer (someone who writes code) and Tester. Most roles, however, are not mutually exclusive in this manner – common sense should prevail.

The Inception Phase

By defining the Development Case, we define the structure of the project, the artefacts that will be produced (and hence the activities that will be carried out) and the roles of all the team members. All that needs to be done now is to start the project proper.

The primary objectives of the Inception Phase (taken from RUP) include:

1. *Establishing the project's software scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be in the product and what is not*
2. *Discriminating the critical use cases of the system, the primary scenarios of operation that will drive the major design trade-offs*
3. *Exhibiting, and maybe demonstrating, at least one candidate architecture against some of the primary scenarios*
4. *Estimating the overall cost and schedule for the entire project (and more detailed estimates for the elaboration phase that will immediately follow)*
5. *Estimating potential risks (the sources of unpredictability)*
6. *Preparing the supporting environment for the project*

At the end of this phase, we would hope to be able to say with confidence, “we understand the business problem to be solved”. So let’s look at the above items in a bit more detail.

Step 1 - Establishing the project's scope

Sometimes this will be a straightforward job because a client provides us with a specification document as part of a contract. Other times, we will have to

negotiate with the Customer to agree which features will be included and which excluded from the application. Typically, this will be thrashed out during a Requirements Brainstorming session. The outcome is usually documented in the form of **Stakeholder Needs**.

Step 2 – Determine Critical Use Cases

RUP projects document functional requirements by employing Use Case Analysis. This technique involves gathering requirements in a number of separate stages each involving a more thorough investigation of needs.

- First, we record the names and a brief description of each use case
- Next, we describe some basic use case scenarios or instances
- Then we document some alternate flow that handle errors and more unusual situations
- Finally, we complete the use case by adding all remaining details. We might also supplement the specification by adding other UML models/diagrams.

As we go through this process, we will naturally learn a bit more about some requirements than others. This should not stop us documenting as much detail as possible, regardless of the fact that the resulting use case won't all be finished to the same degree of completion.

Step 3 – Propose Candidate Architecture

During the Inception phase, we won't normally get more than the names and a brief description of each use case. Nevertheless, this should provide the Software Architect with an insight into the riskiest use case or the ones that have most architectural significance. These use cases, or perhaps even one scenario, will be the first use cases to be developed in the Elaboration Phase.

From past projects and experience, Dunstan Thomas uses our own Architectural Design Pattern, called *Primus*, which we always start with. This helps to avoid starting every project from scratch. After we have carried out an initial requirements investigation, we begin to modify the architecture to suit the individual needs of the project. This is a bit like a civil engineering architect assuming that each new building will be built around a steel frame and then modifying it to cope with the specific demands of the Customer e.g. a swimming pool on the roof or escalators on the outside of the building.

Having proposed the architecture, it will be down to the Elaboration Phase to prove that it fulfils the essential functional and non-functional requirements.

Step 4 - Estimating the overall cost and schedule

Customers naturally want to know how much their application will cost and when it will be delivered. Unfortunately, the Inception phase is not a good time to ask – unless we have some hard-and-fast experiences on which to base our estimates. At this stage, it's a bit like being asked to estimate how long it will take to walk from London to Paris – if we've done it before, fine we can provide

an estimate based on experience (of the route, the ferry/train timetables, etc.). However, if we have never built an application like this, the best we can do is base the estimate on RUP prediction of how much time is typically spent in each phase of a project (see Starting a Project, Step 2 – Produce outline plans). Knowing the number and costs of our resources, we might then be able to **guestionimate** a cost.

One word of caution: Historic evidence tells us that it is not possible for a project to have fixed cost/time, fixed functionality and high quality. The reality is that you can have any two of these but the third must be variable.

Step 5 - Estimate potential risks

Driving risk out of a project is a principle feature of a RUP project. We define risk as:

...Whatever may stand in the way of our success or in the way of achieving major milestones

It's the job of the Software Architect to attempt to identify risk in a project and that of the Project Manager to devise a mitigation strategy and to manage risk on a daily basis.

Identify risk is a very subjective affair. Risks rarely show themselves openly, particularly risks of an architectural nature such as those we attempt to recognise at this stage in a project. Naturally, we will always have budget and schedule risks as well as risks associated with the individuals on the project. However, as stated above, it pays to think in terms of variable functionality if price or schedule is fixed.

The key point to remember is that until an executable has been produced, tested and deployed we cannot prove that we have mitigated a specific risk or that our architectural decisions were correct. Doing this is the major objective of the Elaboration phase.

Step 6 - Prepare the supporting environment for the project.

This is all about setting-up the development (as opposed to the client's) environment. It will involve training appropriate skills, setting up configuration management and version control, making sure developers have appropriate tools, etc. It will also include ensuring that we have a quick and automated mechanism for producing interim builds of an application in order to facilitate parallel testing. In fact, anything that smoothes the development, testing and deployment effort.

Elaboration Phase

The goal of the Elaboration Phase (taken from RUP) is to:

Baseline the architecture of the system to provide a stable basis for the bulk of the design and implementation effort in the construction phase.

The architecture evolves out of a consideration of the most significant requirements (those that have a great impact on the architecture of the system) and an assessment of risk. The stability of the architecture is evaluated through one or more architectural prototypes.

At the end of this phase, we would hope to be able to say with confidence, “we understand the solution to the business problem”. The Elaboration phase is the first phase in which we actually develop some software.

In addition to proving the architecture, the Elaboration phase is where we attempt to drive a significant proportion of the risk from the project, sometimes up to 75%. This too is done by physically developing the software associated with the risk.

Before we start writing software, the Project Manager, with help from the Software Architect, will decide how many iterations to run. In a very risky project, it might be as high as three iterations; a lower risk project will probably have one.

Step 1 – Define, Validate and Baseline the Architecture

In the Inception Phase, we gather requirements for the whole project but only in an “inch-deep and a mile-wide” fashion. Nevertheless, once we had done this, the Software Architect chooses certain architecturally sensitive use cases (or even a single scenario in a use case) to develop in the phase in order to prove his/her architecture decision.

This is done by physically developing the functionality defined in the chosen use case. Given no other influences, therefore, our first Elaboration iteration will probably attempt to address the use case containing the highest risk or the one considered to best prove our architectural decisions. This may or may not be functionality that is high on the customer’s priority list. At this stage, this should not influence our decision-making.

So how do we approach developing our first piece of functionality? Just as with any RUP iteration – like a mini-waterfall

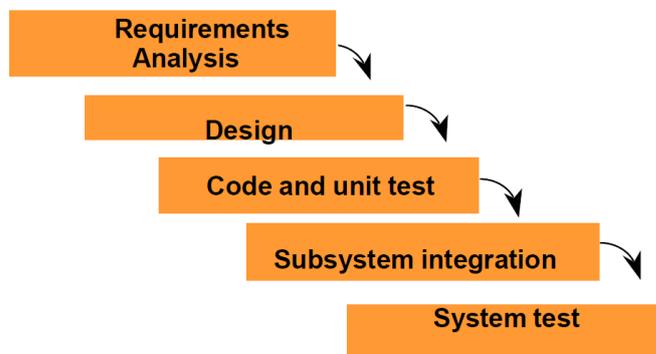


Figure 1 – The Waterfall Development Lifecycle

The end result of every RUP iteration is a testable executable. In this instance, the test would be aimed at proving the architecture and mitigating defined risks. If the project contains significant risk or if we cannot prove the architecture by building a single use case, subsequent Elaboration iterations will be required. Nevertheless, each iteration would be run as a mini-waterfall as shown above.

Step 2 – Refine the Vision

The Vision document contains the justification for the project, from a business perspective, as well as a list of features. Once we have attempted to baseline the architecture, we might have to reduce the scope of the project as a consequence (hopefully, we won't have to). Assuming our architect has been proven and we still believe we can accomplish everything within the stated time and cost parameters, this activity will simply involve sharpening out estimates and plans.

Step 3 – Develop plans for the Construction Phase

At this stage, we will have completed our development activities, proved our architecture, mitigated significant project risks and worked through the whole development lifecycle at least once. This practise will have provided the Project Manager with insight into how accurate or otherwise his team are at estimating effort and how high the overall quality of their work is. Armed with this, (s)he will attempt to construct or refine plans for the next Construction phase of the project.

As mentioned above, meaningful plans can only be based on realistic experience. Hopefully, the Elaboration phase will have provided some of this. If the Project Manager doesn't believe that this has happened, some caveats should accompany his plans and estimates of effort/cost. The key point to remember at this stage is to under-promise and over-deliver.

Step 4 – Refine the Development Case and Development Environment

The very first activity we did when we started the project was to produce a Development Case. At that stage, the required artefacts, project plans and roles were not much better than educated guesses. Therefore, having now gained some real experience, we attempt to refine and baseline the Development Case. The aim here is to cut out any unnecessary work/artefacts and re-organise the team to increase productivity. This might mean selecting new roles or reassigning them or increasing/decreasing the size of the team. It is perfectly normal to have a different shape and size of team for each iteration, after all there are different activities occurring in each requiring different skills.

Finally, based on our development experiences, we might want to implement and recommendation arising from iteration assessments concerning the development environment. For instance, we might consider that the team's OOAD skills are not as good as they could be and hence we arrange for some more training or a mentor. Alternatively, we might have some issues with testing tools. Whatever

it is, this is our chance to improve the development environment before the major bulk of the development begins.

Step 5 – Refine the Architecture and select Components

The final activity is to review the agreed architecture for any further improvements and select any off-the-shelf components we might choose to consider. By using component-based architectures, the buy-or-build decision is always available. For example, Dunstan Thomas's *Primus* architecture employs *Crystal Reports* as it's reporting engine and *ObjectSpark* as it's data services layer. Tests have proved that this alone can reduce the total coding effort by up to 40%, a considerable time and cost saving (contact gstone@dtomas.co.uk for details of these products).

Construction Phase

In most projects, 80-90% of the functionality of an application is built in the Construction Phase. The purpose of this phase (from RUP) is therefore as follows:

The goal of the construction phase is on clarifying the remaining requirements and completing the development of the system based upon the baselined architecture. The construction phase is in some sense a manufacturing process, where emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality. In this sense the management mindset undergoes a transition from the development of intellectual property during inception and elaboration, to the development of deployable products during construction and transition.

At the end of the Construction phase, the whole application is said to be at beta release stage. All that remains is to conduct fully user acceptance testing in the users working environment. This is the purpose of the Transition Phase.

Phase and Iteration Planning

One of the activities we did towards the end of the Elaboration Phase was to refine the Development Case based on our experiences during that phase. This would have involved amending project plans. A key planning issue is to decide how many iterations to run in each phase. Generally speaking, high-risk project require more iterations than low-risk projects, particular with respect to the number of iteration in the Elaboration phase (the riskier the project, the more iterations we run). When it comes to planning the number of iteration in the Construction phase, whilst risk is still an important factor, Customer needs also come into our decision- making. For example, the Customer may demand frequent deliveries in order to report progress to key Stakeholders. Therefore, our plans need to take this into consideration.

In general, most RUP projects will have 6+/-3 iterations. Here's an example of how this might work out for projects containing low and high risks:

Phase	No. Iterations - Low Risk	No. Iterations - High Risk
Inception	0	1
Elaboration	1	3
Construction	1	3
Transition	1	2

Table 5 - Iteration patterns depending upon risk

project when we believe that the project contains more risk. This is because each completed iteration gives us a chance to re-assess progress and risks and re-set objectives for further risk mitigation in the next iteration. For medium- risk project, we could choose any pattern of iterations in between, say 1-2-3-1.

Parallel Iterations

Without doubt, one of the most difficult aspects of managing a RUP project is managing the iterations themselves. We've already seen that the content of each iteration is identical to a standard waterfall lifecycle. From that perspective, an experienced software professional will find RUP very approachable. However, running a project with relatively short iteration brings about some considerable resource planning issues.

The received wisdom is that running parallel iterations should be avoided. Indeed one could argue that if iteration are run in parallel then we are not doing RUP development at all. The major problem with running parallel iterations is that we don't get the chance to learn from one iteration and implement suggestions for improvement in the next.

Discontinuous Activity

Let's take an actual example of a project running according to the RUP lifecycle. Imagine we have a team of 6, each with single skills in the major development disciplines of:

1. Business Modelling
2. Requirement gathering and documentation
3. OOAD
4. Implementation
5. Testing
6. Project and Configuration Management

Let's assume that we are about the start the first of three iterations in the Construction phase. On day one, everyone except the Business Modeller is doing nothing. However, in a day or two they are able to hand-over some of their work so that the Requirements person is able to start to gather and document requirements. Meanwhile, the developers are still doing nothing. By the time the requirements people have something for the OOAD experts to start work, the Business Analysts are probably running out of things to do. Once Implementation is fully underway, the Business Analysts will need something constructive to do. However, as soon as the Requirements people have something to deliver to the OOAD people, they can begin to design some tests based on the functional and non-functional requirements. The moment the OOAD have completed anything,

they can pass it onto the Developers for Implementation, by which time the Testers should have something concrete for them to test against. In the meantime, the Project Manager will have been busy throughout (no peace for the wicked!).

This clearly begs the question: *What do we do with those resources that haven't yet started their work or who have just finished it?*

Should we have them sitting around doing nothing?

If our iteration plans has several use case in it, obviously they can begin work on their part of the next use case. But what if there is only one or they have just finished the last use case in the iteration? Of course the temptation is to get them working on the next use case regardless of whether it happens to be in the next iteration. However, as we noted above, RUP frowns on parallel iterations...

Practical Solutions?

The best way to avoid this problem is to have resources that are multi-skilled, capable of carried out all the RUP disciplines with equal skill. That way, each resource works on their own use cases from beginning to end. Nevertheless, we are still going to get some resources finishing before others. In this case, where possible, they should help those who haven't finished. Whilst this would keep them busy, it might not, in reality, speed development up.

But is this really *realistic* ? Is it possible to have a whole team with equal depth of skill in all the RUP disciplines?

I suggest not. However, it might be possible for some resources to do compatible activities, such as requirements gathering and testing or perhaps Business Analysis and OOAD. By choosing compatible activities, it should be possible to minimise the time individuals spend between jobs. However, the larger the team, the more difficult this gets.

Iterative Development - The Goal

- Try to run each iteration as discretely as possible with minimum overlap between them
- If it becomes necessary to run parallel iterations, conduct a premature iteration assessment prior to starting the new iteration implementing recommendation derived from the current iteration
- Encourage all team members to learn new disciplines – the more multiskilled and flexible they have, the fewer resource management problems you will have
- Where practical, pair-up compatible disciplines for one person to carry out
- Never neglect to conduct iteration assessments and always build in resulting suggestions for improvement into the next iteration plan

- Focus the team's attention on delivering software rather than on finishing all the tasks³

Transition Phase

The final phase of a RUP project is the Transition Phase. The primary objectives (from RUP) of the Transition Phase are:

- beta testing to validate the new system against user expectations
- beta testing and parallel operation relative to a legacy system that it's replacing
- converting operational databases
- training of users and maintainers
- roll-out to the marketing, distribution and sales forces
- deployment-specific engineering such as cutover, commercial packaging and production, sales roll-out, field personnel training
- tuning activities such as bug fixing, enhancement for performance and usability
- assessment of the deployment baselines against the complete vision and the acceptance criteria for the product
- achieving user self-supportability
- achieving stakeholder concurrence that deployment baselines are complete
- achieving stakeholder concurrence that deployment baselines are consistent with the evaluation criteria of the vision

The key to success in this phase is planning and expectation setting. We mentioned earlier that we should under-promise and over-deliver. Here's the time to do it.

In order to make the most of this phase, Customers need to be ready, willing and able to test their new application. This means that:

- A full test environment should have been set up
- Plenty of realistic test data should be available
- Documented user acceptance criteria should be available
- Sufficient time should be available to conduct a full suite of functional and non-functional tests against the acceptance criteria

In addition to this, the Customer's expectations need to have been set to *expect* some defects. This is vitally important. Conversely, the development team should anticipate some last minute change requests.

One way to cope with this to plan two iterations in the Transition phase:

- In the first iteration we carry out exhaustive testing and we gather any last minute change requests

³ The danger here is that if someone gets carried away with their pet discipline and wants to finish all they have to do, we end up reverting to a project lifecycle that more closely resembles waterfall. Each discipline needs only to be completed sufficiently to allow the next discipline to begin.

- In the second iteration we implement all fixes to defects and all last minute change requests

Any subsequent change requests should, if practical, be moved into "version 2". Naturally, there won't be any further defects!

Conclusion

Dunstan Thomas have been using, teaching and mentoring RUP since 1999. If you have any further questions about it, feel free to contact Graham Stone (gstone@dthomas.co.uk).