The RUPple Effect

Richard Tanner-Tremaine, Dunstan Thomas Consulting
http://consulting.dthomas.co.uk

RUP is just one of many acronyms and buzzwords that can strike fear into the heart of even the bravest souls in the software development world. Indeed, any mention of using a "Process" with a capital "P" when referring to software development projects makes the finance department hide the cheque-books, send development managers into a panic over the wasted time their developers will spend on documentation instead of writing lines of code, and developers start thinking about the best way to word their resignation letter. Yet we all know that these "Processes" are supposed to be a "Good Thing" … so why the unease?

Misconception and Misdirection

A lot of our sense of mistrust probably comes from our fear of change. Software developers are notoriously egotistical – that's part of what makes us so good at what we do.

We all know, without a doubt, that we are the best at what we do, our ideas are the right ones, the way we work is the most productive possible… and we can prove it (and usually do)! We're so used to working in a certain way that having to conform to some new standard and slot into a Process like good little sheep instead of just being able to do what we do best is quite disconcerting. Not to mention all the extra administration and paperwork – what a pain!

We've all heard about RUP too. That's that massive collection of documentation tied up in some overblown Process where you spend at least half your time writing documents and making pretty diagrams in UML instead of doing some actual work.

You've probably heard all of that before, and more! This is, despite numerous PR efforts to the contrary, a very common misconception. What's worse is that RUP has become so much of a buzzword that any mention of any sort of Process is often associated with it. Even more of an issue is the association between RUP (and hence any Process) and the tools Rational Software provides.

Sure, if you're serious about adopting the Rational Unified Process in it's entirety, you might want to consider using Rational Rose for your UML modelling, Clearcase for Source Control, and Requisite Pro for requirements management. You might even want to use Rational Robot for automating some of your testing. But these are not an actual requirement!

You could follow RUP as your process, but instead of using the "Industrial Strength" Rational tools, use ones you've already got, or even go for cheaper options (which are sometimes better for your uses). I've worked on projects which follow RUP as the core of their development process, but have not used a single Rational tool. Enterprise Architect from Sparx Systems is a popular choice for the UML modelling. VSS, PVCS, Starteam – you're probably already using one of these for source control, so unless it's causing problems, why change? Calibre RM from Borland, and even the RaQuest plug-in for Enterprise Architect might suit your needs as far as requirements management is concerned (or any number of other offerings). And perhaps D-Unit / J-Unit or some other test harness is more than satisfactory for your requirements when it comes to testing. RUP as a Process does not necessarily mean to have to go out and buy lots of other Rational products! RUP is a process – it's not a set of tools.

Hopefully the above will put your finance department at ease!

RUP as a Software Development Process

Now let's get back to tackling RUP as a Process. There are, to my mind at least, a few key aspects of RUP on which the rest of it is built. The first thing to realise is that RUP is an iterative approach. Secondly, RUP defines roles and responsibilities, which are then assigned to people involved in the project. And thirdly, RUP has a road-map.

An iterative approach is simply breaking down a project into manageable chunks (iterations), the idea being that at the end of each chunk of development effort, you have something to show for it, and will have to opportunity to review what you've achieved, decide whether to carry on, and also – and this is the important bit that's so often overlooked – review and adapt the Process used in the iteration
.
The roles and responsibilities in RUP are an attempt by Rational to identify and name the types of activities people do, and what they are responsible for in a software development project. One person on the project might actually perform several roles (and hence have the associated responsibilities), just as one role might be shared amongst many people. It is not strictly a 1 to 1 relationship.

Rational then took the concepts of iterations, roles and responsibilities and came up with what is essentially a big flow-chart showing what the people taking on the roles can expect to be doing during an iteration – what I refer to as Rational's "road-map to software development". They also decided to make our lives a little easier by providing us with sample templates for documentation purposes when documentation might be required. Some of the documentation is UML-based and some is textual documents.

Well, there's obviously a lot more to RUP than just the above. It is, after all, an extremely detailed study in software development methodology and processes. The point I'm trying to make is that RUP is essentially an Iterative Process that follows some best practices, and provides a detailed template for your company to base their own software development process on.

That's right… you did read that last sentence correctly. RUP is a **template** that you can use to base your own process on. RUP is a process framework which leads to an Iterative Process, and part of each iteration is actually reviewing the Process! This is something that is often overlooked or is not understood. Rational does not recommend that you follow their process in its entirety. They provide a framework which you're supposed to tailor to your individual needs. As with any good Iterative Process, improvement and adaptation of the Process to suit the needs of the people using it is encouraged.

Thinking about adopting RUP as the basis of your own software development Process? Then instead of just jumping straight in, take some time to consider what parts of RUP are actually going to help, and what are likely to hinder. You don't think you need to produce a particular document RUP suggests? Then don't. They key is to make sure that everyone is on the same page. In other words, make sure you document your Process! It is not the individual people who get to decide whether or not to produce a particular document – it's the Process that guides that decision. But hopefully when you review the Process at the end of an iteration, whether or not to produce that document can come up for discussion.

So the key thing I'm trying to point out here is that you can, and should, adopt the Process according to your own needs and experiences. Whether you're following RUP or some other form of Iterative Process, make sure that at the end of each iteration you have your reviews. Get everyone involved:

management, developers, customers, anyone who has a stake in the project. It doesn't have to be all in one massive meeting – just make sure you get feedback from everyone so you can move forward having had input from all concerned.

Another misconception when it comes to documentation in an Iterative Process s that when you start writing up a document, you need to finish it. This is true, but not in the way you're probably thinking. Documents should be created or updated to such a point that it is possible to use them to carry on with the Iteration. This applies to textual documents as well as UML or other forms of documentation.

Why spend days writing up a complete specification for a subsystem when you know you're only actually going to use the first 10% for the next iteration? Chances are that after the iteration you'd have to go back and change the rest of the document anyway, so why waste time now? Likewise, why bother creating 6 different types of UML diagram down to miniscule details when a class and sequence diagram with enough information on them to convey the necessary information to the developer for the current iteration will do? The point is this: you can always come back and update the documents when you need to, so why do it when you don't need to?

It's an iterative approach! Do what you need to when you need to – that way you'll manage change in the project a lot better. You're far less likely to waste time writing documents that end up never being used when the customer changes their mind, or you hit a snag in development.

To RUP or not to RUP

So… is RUP for you? Maybe. If you have a project team larger than five developers, then chances are you will actually benefit from a formal and all-encompassing Process. RUP would be a good foundation for you to create your own Process on.

If RUP isn't an option as far as you're concerned, then that doesn't mean you shouldn't be trying to define the software development process you use! As I said before – RUP is a foundation for a Process. This does NOT mean that all Processes are RUP.

In addition to RUP there are lots of development methodologies out there to consider. XP (eXtreme Programming), Scrum, FDD (feature-driven development), and a bunch of other agile methods all talk about how your development team should work to maximise their effectiveness. They will all try and convert you to their way of thinking, some of their followers will too in an almost zealous manner! The truth of the matter is that there is no single methodology that will work for each and every team on each and every project. There are far too many factors involved to be able to prescribe a single solution to every problem.

Personally, I feel that an iterative approach to software development is definitely the way to go as far as a Process is concerned. Not only does it give you a chance to keep all stakeholders involved throughout the life of the project, but it helps you manage change a lot better: change in requirements, change in process, change in methodologies, change in scope, change in finances, change, change, change. All are dealt with a lot easier if you're only worrying about small chunks of development at a time.

Couple an iterative approach to your favoured methodologies as you see fit. Try XP for a few iterations if you're keen. If it doesn't suit your environment, keep what you like from it and fill in the gaps from another method. The key to success is being able to adapt. Adapt the way you work, not only what you're working on.

The key to any Iterative Process is to have the reviews. Make sure you review everything – including the Process. Remember to only do what's necessary to complete the next iteration. Writing up a long document on what the optional built-in media player will look like in the first iteration when you have no idea whether the client is actually interested is, quite simply, a waste of time.

And most importantly … don't expect to implement a new Process over night. People don't like big changes, break it to them gently. Get them using new tools one by one. Try out your new ideas for the Process on smaller projects or in small teams first. That way, if there are any problems, you get a chance to remedy them without getting everyone up in arms right away. A lot of the resistance to adopting a new Process (RUP or otherwise) is because of the expectation that it's all going to happen over night and nobody is going to know what he or she is supposed to be doing. Do it in an iterative fashion, review each new thing you introduce and getting your new Process in place will happen as a matter of course.

Conclusion

RUP is a Process Framework – it is not a set of tools.
RUP is a Process Framework – not all Processes are RUP.
RUP is a Process Framework – adapt it to your needs, that's part of the Process.

Unfortunately, there is a lot of misunderstanding when it comes to the software development process. Whether it's just good marketing on the part of Rational, or simply a lack of understanding and resistance to change when it comes to the unknown, this automatic association is actually an unfortunate thing for everyone involved in software development. Not only does it make it tricky to introduce new Processes and refinements to development methods as far as the developers and their managers are concerned, it makes it a hard sell to customers too.

Don't get me wrong – RUP is great! But like anything, it has its place. It's just really unfortunate that any time someone mentions "UML" and "Process" in the same sentence that everyone thinks you're trying to shove the whole of RUP down their throat. We need to get across that RUP is a template – somewhere to start from – and it's not something we're alluding to every time we talk about software development process and UML.

Essentially, we need to smooth out the waters again so that everyone concerned can see their reflections a bit more clearly without all those RUPples. Let's get across that having a defined Process is good. That having an Iterative Process is better and that following best practices along the way with UML and OOAD as the core will give any project a head start.