BASIC CONCEPTS: Integrating Rational Unified Process into Traditional Development Methodology

Introduction

This document is based on experience as part of a Consultancy team that provided assistance to a client who wanted to integrate best practice from the Rational Unified Process ("RUP") into their more traditional development methodology, based on more traditional waterfall development (similar to "SSADM").

The exercise of integrating these two approaches can be complex and involve many different teams within the organisation, including:

- Key Development Managers, in terms of deployment and take-up of this approach.
- Project and Programme Office, in respect of reporting changes.
- Quality Assurance teams, who are often "guardians" of the methodology
- IT Production, who will need to understand the different Infrastructure demands caused by an iterative development approach.

This document cannot cover all of the aspects involved. Instead, the objective is to outline the basic concepts that need to be addressed in integrating RUP with, say, SSADM.

What is RUP, and how does it differ from SSADM?

The Rational Unified Process (RUP) is an iterative software development process created by the Rational Software Corporation, now a division of IBM. The RUP is not a single concrete prescriptive process, but rather an adaptable process framework. In practice, the best way to use RUP is to either (1) tailor it using the "RUP Builder" utility, and create your own methodology based on RUP, or (2) start with your existing methodology and build RUP-like components into it.

There are a number of key differentiators between RUP and a more traditional Approach.

RUP has a strong Architectural Focus

One of the key aspects of RUP is that it focuses on the Architecture Design of the proposed solution. This is much more relevant when designing modern web-based applications which need to be hosted on multiple servers.

RUP is Risk-Driven

With its emphasis on the Technical Architecture of the proposed development, RUP focuses on the key Architectural Risks that are associated with the proposed design. The Architectural Risk Log is therefore a key mandatory part of the RUP approach. It is used to determine how to plan the iterations of the project.

RUP is Iterative

The main part of RUP development ("Elaboration") is iterative. The purpose of each Iteration is to focus on one of the significant Architectural Risks, and putting in place a final developed solution to mitigate that risk.

It is this Risk-driven, Iterative, Architecture approach with characterises RUP.

RUP Phases

A RUP project therefore consists of the following phases:

- Inception – starting the project, and identifying the Architectural Risks
- Elaboration – this is the iterative part of the project, where each Architectural Risk is iteratively addressed in turn.
- Construction – responsible for integrating these iterative phases together, and creating the final product.
- Transition – deployment of the application to live.

RUP has an emphasis on using UML, Sequence Diagrams, State Diagrams and modelling tools to describe the application being developed. This means that RUP is typically deployed alongside development tools such as Requisite Pro (for requirements gathering) and Rational Software Architect ("RSA"), also available from IBM. However, it is not essential to have the tools in order to use the approach.

In addition to the above, there are a number of aspects of RUP which can be considered "best practices", and which can be readily incorporated into other development methodologies, e.g.

1. Develop software iteratively
2. Manage requirements
3. Use component-based architecture
4. Visually model software
5. Verify software quality
6. Control changes to software

Where is it beneficial to incorporate RUP?

Since RUP is an Architecture-Driven methodology, it provides real value when developing modern web-based software, where the physical and logical topologies are more complex.

RUP can be beneficial when the company wishes to introduce new Architectural developments.

RUP effectively provides a formal framework for the "proof of concept" which is often needed when developing new complex applications.

By it's very nature, RUP addresses some of the "high-risk" development projects that a company may wish to initiate.

Why integrate these two approaches?

Typically, adding RUP-like aspects to a tradition development environment has a number of benefits. For example, it:

- enables developers to transition from traditional to more modern development more easily.
- preserves the existing investment in methodology know-how within the development community
- facilitates cross-fertilisation of ideas between existing and new developers]
- eases the transition for IT Production and Deployment.

Challenges of RUP / SSADM Integration

Integrating RUP aspects into an existing Development Methodology has a number of significant challenges.

The most significant of these is the Cultural Challenge. To be successful, developers need to be able to identify the Architectural Risks associated with a project, and use these to drive the iterative process. In my opinion, it is even more important to have good strong Project Control and Governance than in more traditional design. This is because individual Iterations (during the "Elaboration") phase need to be carefully controlled with formal iteration objectives and success criteria.

At the same time, introducing RUP often involves the introduction of Object-Orientated development languages and tools. This also provides a training challenge, since the concepts of OO development, if implemented properly, are very different from more traditional procedural development.

RUP also emphasises a visual approach to design, which is best implemented by introducing UML and other techniques. These also have a steep learning curve.

Taking the Pragmatic Approach

Both RUP and SSADM can have significant overheads on a medium-size project, particularly if implemented in "vanilla" form.

It is essential to take the key aspects which will benefit the organisation.

Some questions that may need to be asked are:

- What documentation is essential in order to proceed to the next stage / phase, given the existing development experience, and controls currently in place?
- Is the end-goal of each document fully understood?

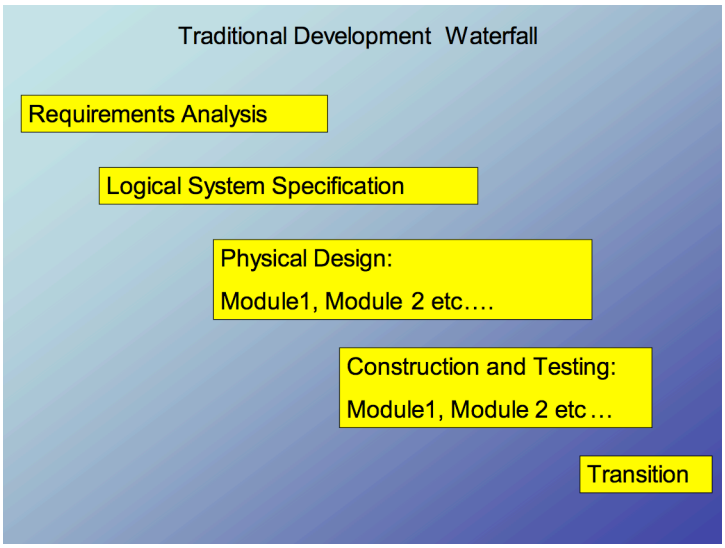
Most development projects that I have seen over the last 20 years (mainly in the private, rather than the public sector) mandate only a proportion of the possible deliverables (or "artefacts", to use RUP's terminology) that the full methodology would require.

The key decision is which artefacts are significant in determining the progress of the project.

In addition, the learning curve for a new methodology should not be under-estimated. In my opinion, it is more important to allow developers to grasp the overall concepts at the early stage, rather than impose huge amounts of additional deliverables, which will only serve to bring the methodology into disrepute.

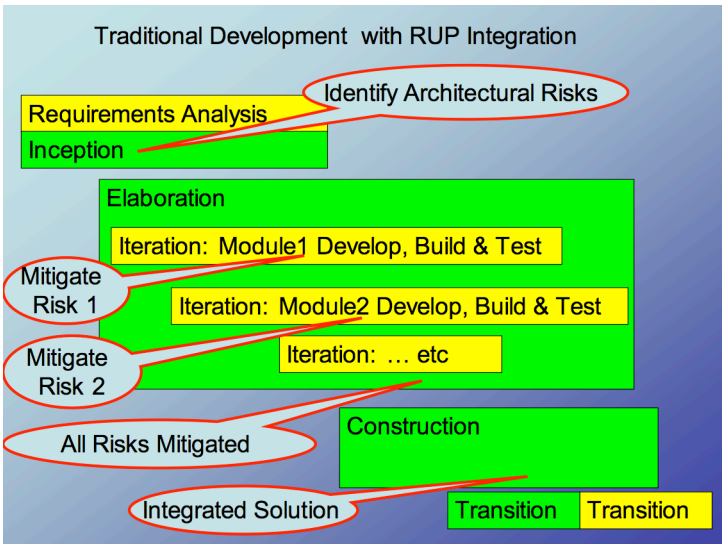
Traditional Development Methodology

The following diagram shows a (very) high-level view of the main time-based stages involved in a traditional SSADM type project. We have not included any of the controls, roles and responsibilities of the project. The purpose of this diagram is to show the traditional "waterfall" of development over time. For ease of understanding, a number of stages have been removed.



A "RUP-View" Development Methodology

Taking the above time-chart, which is very basic, it is possible "RUP-Enable" it to show how some iterative development, based on architectural risks, replaces the previous stages.



This is by no means the only way to RUP-enable the methodology, but it does provide an understandable frame of reference for users of existing methodologies.

Impact on IT Production

It is sometimes forgotten that the introduction of a RUP approach can have a major impact on the Infrastructure and Production side of IT.

New "Architecture Driven" projects typically have a large element of IT infrastructure. This leads to the introduction of a new Architecture role – Infrastructure Architecture (or "Production Architecture").

The traditional split between Application Design and Infrastructure Design tends to become blurred. As a result, it makes sense for the early stages of a project to become much more Collaborative between IT Application Development and IT Infrastructure. Decisions taken on Application Design can have significant impact on Infrastructure Requirements, and vice-versa.

It is no longer the case that "Developers create the application and Infrastructure hosts it" – since Application decisions are not "Infrastructure-neutral". Choices of messaging systems, protocols, transaction control, cache controls etc. all have significant Infrastructure impact.

There should be an ongoing collaboration between the two architecture teams (Applications and Infrastructure). Failure to facilitate this collaboration can lead to project disaster, or an un-deployable application.

Also, the introduction of an Iterative approach fuels the demand for additional deployments of Infrastructure, in different time-scales. Instead of deploying all the Infrastructure at the deployment phase of a Project, IT Infrastructure need to be able to provide piecemeal chunks of Infrastructure, for each Iteration. This can have a major impact on the way in which IT Production is managed and controlled.

Conclusions

It is possible to introduce RUP concepts into a Traditional development environment in a phased, gradual approach, without resorting to a "big-bang" change to working practices.

Nevertheless, introducing RUP is a major initiative, since it is:

- Architectural-Based, as distinct from Design-Based
- Risk-Driven, as distinct from Process-Driven
- Iterative, in order to mitigate Architectural Risks

In addition, this often coincides with the introduction of Object-Orientated Languages and Visual Development tools, which add to the learning curve.

Successful introduction also requires input from many additional teams, including Quality Assurance, Project and Programme Office, as well as IT Production.

Nevertheless, it is possible to put in place processes that build on the Client's existing working knowledge and enable a transition to a more modern development methodology.

Sources

- RUP itself if available from IBM Rational. See the IBM web site at http://www-306.ibm.com/software/awdtools/rup/

- For more information on RUP, there is an excellent WIKI site, at http://en.wikipedia.org/wiki/Rational_Unified_Process

- As a basic recommendation to SSADM, and it's approaches, and concepts, see http://www.comp.glam.ac.uk/pages/staff/tdhutchings/chapter4.html

---

About the Author: Dennis Adams has over 20 years experience with Managers of IT Production Systems, and now provides a specialised Consultancy Service to help IT Production Managers establish Pro-Active and more Client-Focused teams. He can be contacted via his company website at http://www.dennisadams.net