OpenUP –the best of two worlds

Bjorn Gustafsson bjorn@goodsoftware.ca
GOOD Software Inc www.goodsoftware.ca

Abstract

Software organizations looking to adopt an iterative and incremental process have found themselves left with less than ideal options. While RUP, the IBM Rational Unified Process®, was the first mainstream iterative software process, its complexity and size makes it difficult to adopt. Agile processes like Scrum and XP, on the other hand, are leaner, but their different culture and lack of documentation often meet resistance from management.

This dilemma is perfect soil for the new OpenUP process which packages the best RUP and agile practices in a light-weight open source process framework. The result makes management happy, since they get a stable and well-defined governance process, is easy to adopt, and serves the team a smorgasbord of software best practices right in their web browsers.

This article gives an overview of OpenUP and explains how it relates to both RUP, from which it received its foundation, and agile methods, from which it incorporates their best practices.

Introduction

Software development today is radically different than it was only ten years ago, and the software process landscape is changing too.

Since its creation ten years ago, RUP [11] has become the de facto process in many software organizations. With a knowledge base of thousands of pages it offers guidance to a wide range of industries and systems. However, RUP is also "complex" and can be difficult to implement in an organization because of its size, complexity and learning curve.

More recently, enabled and fuelled by new conditions and initiatives in our industry, the Agile Manifesto [2] established a philosophy in 2001 that brought some ground-breaking new ideas to the software process landscape, which materialized in methods like XP [3] and Scrum [4]. Even though replacing RUP never was the primary motivation, 'agility' seemed like a good idea to many, including those that struggled with RUP.

Now, a few years later, we know that agility is not the cure-all for our software process pains, and software teams still keep failing. "Being agile" requires a change of mindset and attitudes throughout the whole organization and not all organizations are ready for this cultural change.

Some common problems can be observed in troubled projects:

- **The project team doesn't share a clear vision of how the system will appear to its users.** Without a clear vision of the final system, there is no guiding framework for the work in the project. The team's analysts have no means to calibrate their requirements to the scope and effort of the project,

which results in ill-fitting requirements statements; and the development team can not properly prioritize their work.

- **Requirements do not drive development work**. Some development cultures regard requirements as "incidental" input to the project only, and drive the development work based on other, internal and technical, conditions. This is commonly found in "silo" organizations where there are separate teams for requirements capture and development.
- **The system's architecture has not been articulated**. Although projects that only evolve and maintain existing systems may not need to pay much attention to architecture, there are many projects that do. As Dean Leffingwell [7] points out: "*... how much architecture a team needs depends in large part on what the team is building"*.
- **Plans are not connected to the engineering reality**. Plans are often created and maintained separately from the actual project work. We have all seen nebulous Gantt charts that project managers spent days or even weeks creating, with hundreds of line items in nifty breakdown structures, purportedly believed to bring the project to "completion" at some well-defined point in time. Of course, these plans become outdated even before they are pinned on the wall.
- **Risks are ignored**. All projects run the risk of building the wrong product or not being able to build the product as envisioned, yet very few projects acknowledge this uncertainty and actively try to reduce it.

Whether your process is RUP, agile, or something else, and whether you are a programmer, architect, designer, tester, analyst or manager, you may recognize these problems in your own project. If you do, you are not alone. In fact, the majority of software projects are still problem-riddled.

Why is this?

First and foremost, software development is complex matter and orchestrating dozens or more individuals to build a complete software system is down-right hard work. Every project is unique and most projects have some parameters that just aren't "ideal" for their process or they have a less than ideal process.

Many software organizations ask "How can we fix these problems?"

Both RUP and agile methods certainly have the solutions. The problem with RUP, though, is that they can be hard to find and put to practice; the problem with agile methods is that their advice and guidance can be difficult to translate to a particular project situation since they are based on tacit knowledge and textbooks only.

With OpenUP the situation is different. It packs short and concise guidance into a small number of pages, which are always just a couple of clicks away. Adopting OpenUP takes you a long way towards solving these and other problems.

Introducing OpenUP and the Eclipse Process Framework

While OpenUP is the tangible process product, it is also part of the larger Eclipse Process Framework.

> *"The Eclipse Process Framework (EPF) aims at producing a customizable software process engineering framework, with*

exemplary process content and tools, supporting a broad variety of project types and development styles" [1]

EPF is an open-source initiative that was started in 2006 with contributions by IBM Rational of parts of their RUP content and technology. Since its inception the project has actively involved more than 20 companies, and the first release was made available in September of 2007. Despite being an Eclipse project, EPF can be used to create process descriptions for any type of development, including J2EE on Eclipse or .NET using Microsoft Visual Studio.

EPF, like other Eclipse projects, offers exemplary implementations of its two components:

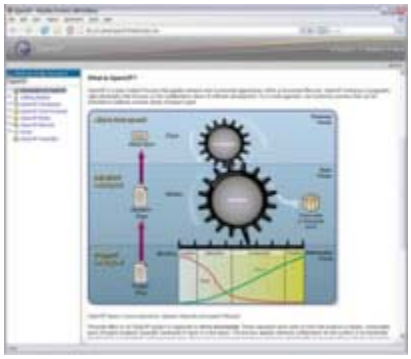

Figure 1: OpenUP process

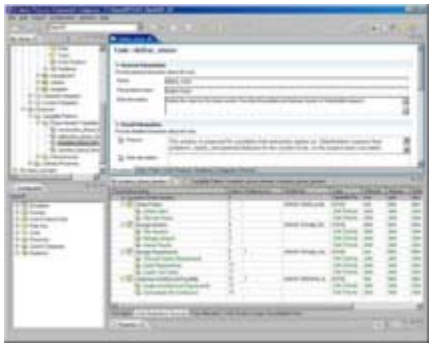

Figure 2: EPF Composer

OpenUP is a welcome addition to the software process landscape: it is agile, both in the guidance it provides and in "spirit", while at the same time being documented. As Per Kroll, the EPF project lead states "*the overarching goal with OpenUP was to create an agile approach to using RUP, while at the same time leverage all the good things we liked in other agile processes.*" [6]

OpenUP borrows many strokes from RUP which it blends with agile practices, and the result is a complete process that suites many smaller-scale projects. It can also be extended for those projects and organizations that have more challenging circumstances.

Even if the OpenUP process is the focal point of EPF, it is only half the story. Much of its value lies in the fact that it is **open source** and that it is based on a **standardized meta-model**, called SPEM [5].

The advantages of open source are obvious: it can be used by organizations of all sizes at no cost. It also means that anyone can share their best practices in the spirit of open source.

The advantage of being based on a standardized meta-model may not be as obvious, but is key to the overall success of EPF. Concretely, this makes OpenUP **extensible** and it can be augmented with any combination of standard, 3rd-party and proprietary practices. Just like UML provides a standard language for software design models, SPEM provides a standard language for definition and exchange of process models. With SPEM, process is expressed as a set of elements and components, and it is modular as opposed to a monolithic whole.

This is where EPF Composer comes into the picture. It provides the tools both to create new process content and to tailor OpenUP by selecting the desired set of components. It is a fully-featured process engineering tool and any organization can

use it to develop their own process best practices, as well as share those with other EPF users.

This is the "grand vision" of EPF: to build an open source community around EPF, where industry best practices can be downloaded and exchanged. With this first release, this grand vision is starting to come into existence.

OpenUP – the method

This chapter is intended to give a brief introduction to OpenUP only. There are excellent resources on the EPF web site [1] for learning more about it. (including downloading and browsing the OpenUP process itself!)

OpenUP defines a set of **roles, work products** and **tasks:**

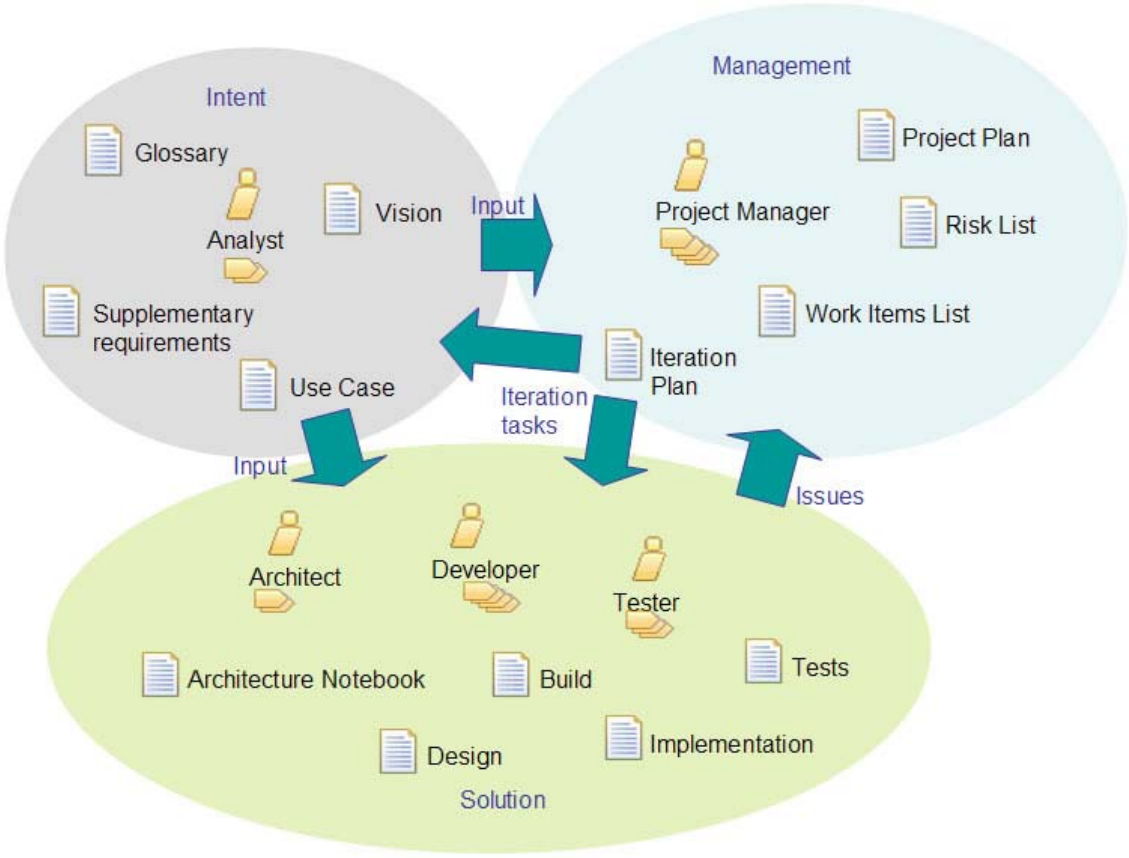

Figure 3: Overview of OpenUP elements

The process described by these elements is *minimal* and *complete;* it is the smallest set of elements that still describe a project end-to-end.

Analysts formulate the intent of the system in the vision, use cases and supplementary specification products.

Managing iterative projects is largely a matter of orchestrating the activities of the team through each iteration, primarily by managing tasks against the current iteration plan. It is populated with the highest prioritized items from the work items list and risk list, and the team commits to a certain amount of work in each iteration.

The development team includes architects, developers and testers, who are responsible for the development of the solution products, which ultimately result in the operational system.

Activity models describe typical task collaborations as they occur in iterations:

Figure 4: Example activity model

A lifecycle model provides the governance process for iterations and micro-increments:

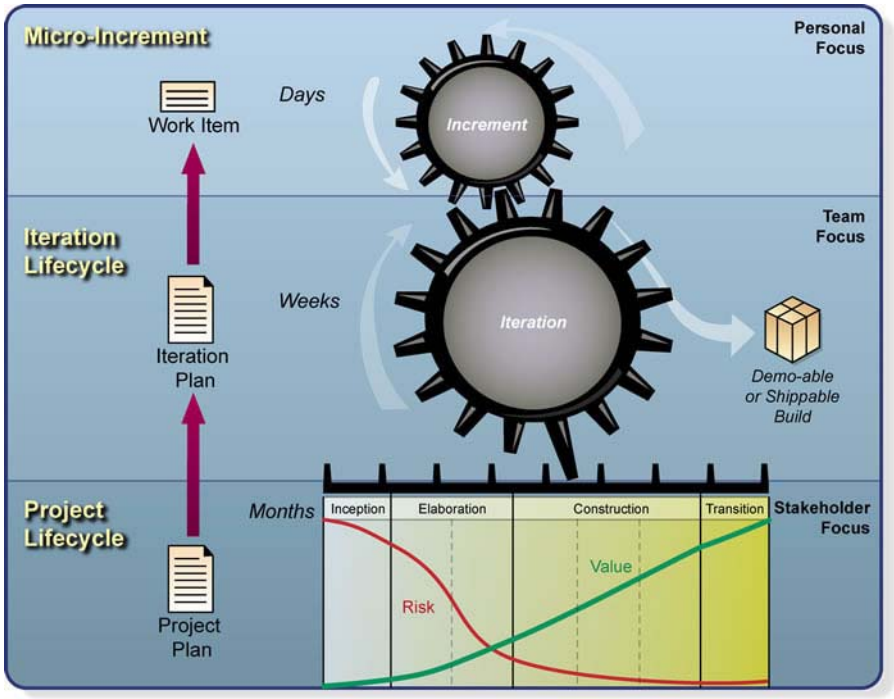

Figure 5: OpenUP lifecycle model

If you are a developer your process is the "analyze-code–test–integrate" cycle that you go through almost on a daily basis, as specified by the tasks in the iteration plan; if you are a manager, or Scrum Master, your focus is on how the team performs in each iteration, which is estimated and tracked in the iteration plan; and if you are a project stakeholder, you are focused on understanding what the project will deliver and when, as described in the project plan.

OpenUP is inherently *iterative and incremental* and the project is executed over a series of iterations, typically 2-6 weeks in duration.

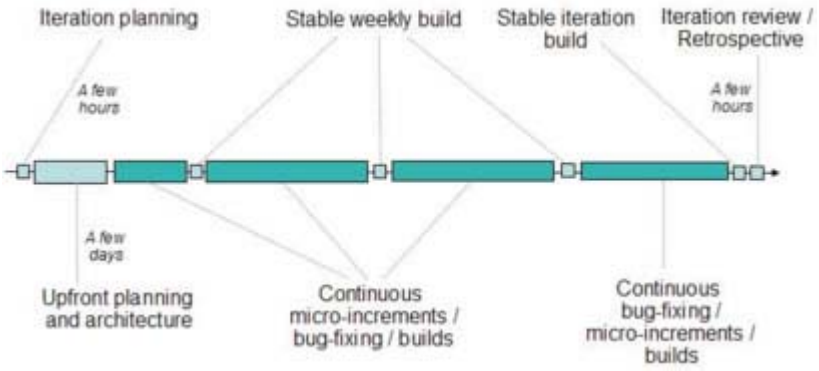

Figure 6: Iteration lifecycle

Each iteration takes on a subset of the project's work items. Iterations are started with a short planning activity where the highest-prioritized items from the work items list and risk list are allocated to the iteration plan. This is followed by a short activity where the team gets involved in the detailed planning and estimation of each work item. Thereafter the iteration work commences and each feature is analyzed, designed, implemented, tested and integrated in its own micro-increment. As the iteration unfolds, task status is reported back to the iteration plan for overall project status.

The project lifecycle in Figure 5 identifies four distinct phases, each with a specific purpose and milestone criteria:

- **Inception**: define the scope and objectives of the project
- **Elaboration**: establish an understanding of the requirements and create an *executable architecture* for key scenarios and quality demands
- **Construction**: build the functionality of the system
- **Transition**: release the system to the end users

This lifecycle model distinguishes OpenUP from agile methods and allows us to focus early project efforts, in the inception and elaboration phases, on understanding the scope of the project and its solution before embarking on full-scale development in the construction phase. At the end of the elaboration phase we have typically spent only 20-25% of the total project budget over 30-40% of the project schedule.

With this brief introduction, let's see how OpenUP helps address the specific problems that we identified earlier:

- **A shared vision is created in the inception phase**. The stakeholders' key needs and features are captured in the Vision document. It describes high-level requirements and design constraints, and gives an overview of the system's functional scope.
- **All project work is driven by use cases and other requirements**. The Work Items List constitutes a "laundry list" of features, requirements and change requests raised on the system.
- **An executable architecture is created in the elaboration phase**. The Architecture Notebook along with the other development work products represent a base-lined executable architecture that demonstrates how the

system supports the key scenarios and constraints, and which serves as basis for the ensuing construction phase. It is worth noting that 'architecture' is not a separate "thing" – it is the underlying organization and qualities of the system being built – and the executable architecture is just a state of the underlying design, implementation and test products. To create the right focus on the architectural concerns we prioritize those use cases and other requirements during the elaboration phase that involve the highest technical risks, as identified in the risk list.

- **Each iteration is planned "just in time", and the project team is involved in the detailed estimation and planning activities**. For each iteration the Iteration Plan is populated with the highest-prioritized work items from the work items list and risk list. The team is responsible for identifying and estimating tasks for each work item.
- **Risks are pro-actively identified and mitigated**. The Risk List identifies a prioritized list of risks that are associated with the project. All critical risks have been removed at the end of the elaboration phase.

OpenUP thus helps us remove some of the main obstacles to project success early in the project and with only a small investment.

The inception and elaboration phases also help establish and organize the key project work products and processes, so that construction can commence with a growing team and aggressive timelines with minimum friction.

OpenUP is a web-based knowledge base

OpenUP is installed in your team's intranet and accessed using regular web browsers. It can be installed directly from the EPF website or, in the case you wish to create your own tailored variant, with the help of EPF Composer.

If you are familiar with RUP, you will recognize OpenUP's content browser and some of its content. If you are not familiar with RUP, you will find that OpenUP's two-pane view and content are easy to understand and navigate:

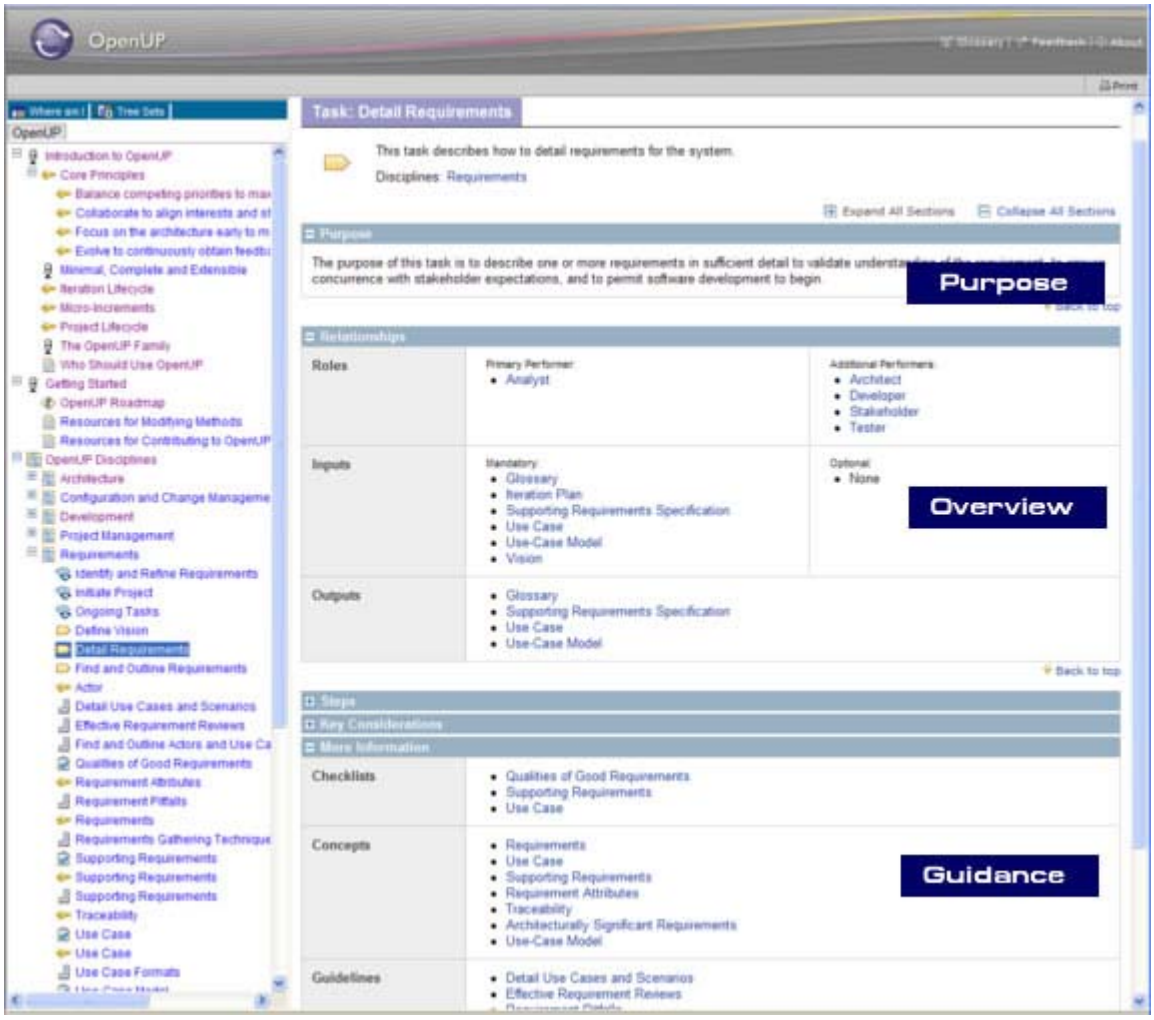

Figure 7: A sample OpenUP page

Figure 7 shows a *task* description and demonstrates the general layout of pages in OpenUP. Each element (*task, role, work product*) has a short description and links to its associated elements.

Each element has *guidance* attached to it, which further explains and facilitates their use in the project:

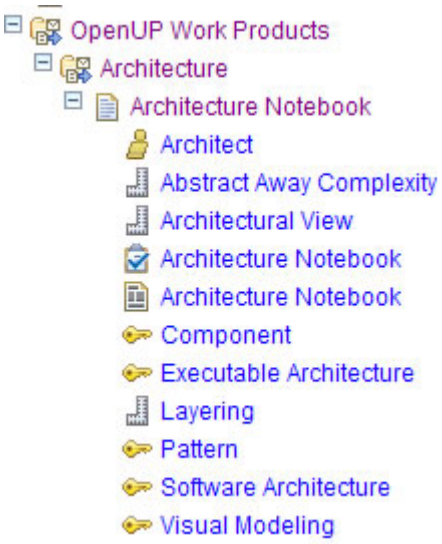

Figure 8: Example of guidance

The process web site provides a complete encyclopedia of your software process, and to understand OpenUP is a simple matter of browsing the OpenUP web site and studying those areas that are of immediate concern for the role you are playing in your project and the work products that you are responsible for.

Tailoring OpenUP to your needs

OpenUP can be tailored and extended using the EPF Composer tool:

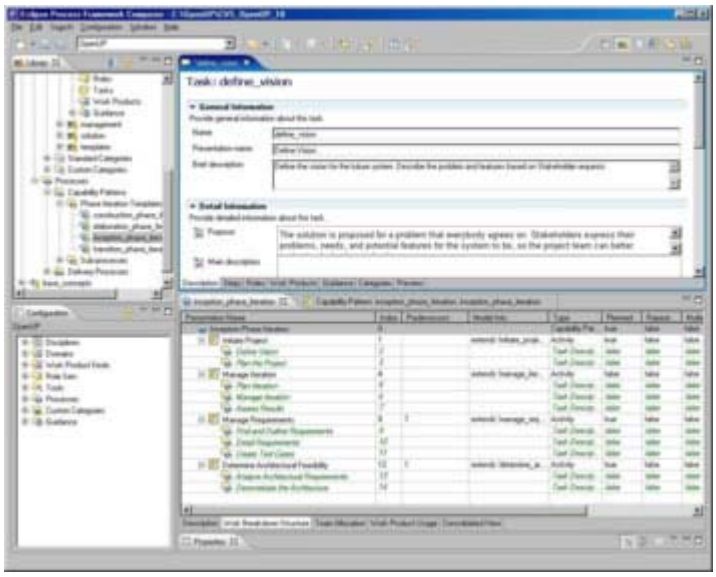

Figure 9:EPF Composer

It is centered around a form-based editor which allows you to quickly create and modify content. Your content can be as simple as a couple of additional guidance pages, or it can be as complex as adding completely new disciplines with new roles, tasks, work products and activities. You can even build an entirely new process family and ignore the existing OpenUP content completely. (An interesting example of this is the Scrum process that was developed as part of the EPF work: it uses the EPF process browser but is completely tailored to Scrum terminology and concepts)

A simple checkbox interface allows you to integrate your extensions with other OpenUP components in your own OpenUP variant.

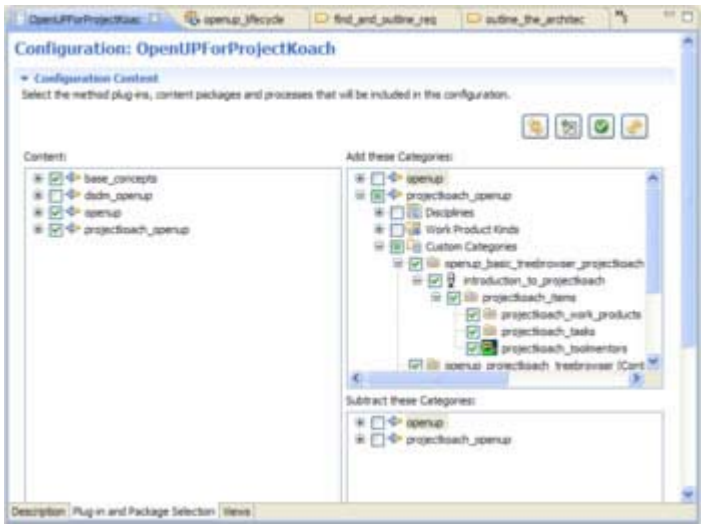

Figure 10:Checkbox selection of OpenUP configuration

Working with OpenUP

Normally, adopting a new software process is a major undertaking, and the transformation of the project team or organization never happens over night. Once a process has been implemented and used you want to find ways to improve it and make the organization more efficient. Process implementation and improvement involves understanding the process, defining it, and training the team on its practices, and this can take weeks, months or even years.

With OpenUP this is not the case. If you want to use it as-is - which is a good starting point for any project - you just download it from the web site and install it in your environment. It is self-contained and provides all the guidance you need to get started. Should you wish to improve the process from there, for example if your project is using a new technology and you want to add guidance specific to that technology, you will find it easy and straight-forward with EPF Composer.

OpenUP has many similarities with RUP, but choosing it is an either-or decision for RUP projects since they can not really co-exist in the same environment (and it wouldn't make sense because of their overlap). A goal with OpenUP was to create an agile process with focus only on the core software development element. This makes it manageable and easy to grasp.

Adopting OpenUP in agile projects, however, is not an either-or decision. It was designed to fit seamlessly with the work habits of agile projects. As we have seen, it balances agility with "just enough" governance, especially in the early stages of the project, to help establish the project context, fundamental structures and key principles early. Once status quo is reached – that is, when the system has taken shape and the team has found a comfortable iteration pace – OpenUP is no different than most agile methods.

OpenUP is based on four core principles, all of which have direct correspondence with the Agile Manifesto:

OpenUP Key Principle	Agile Manifesto
Collaborate to align interests and share understanding	Individuals and interactions over process and tools
Evolve to continuously obtain feedback and improve	Responding to change over following a plan
Balance competing priorities to maximize stakeholder value	Customer collaboration over contract negotiation
Focus on articulating the architecture	Working software over comprehensive documentation

If you are used to working in agile projects you will find that OpenUP is no different when it comes to its underlying values and principles. Nor does it impose a different work style for your daily activities: you still do the daily stand-up meeting; iterations are planned the same way; you design, implement and test in small increments.

If you are a Scrum Master you will find striking similarities between the Scrum product backlog and OpenUP's work items list, and between the sprint backlog and the iteration plan. This is no coincidence since the inspiration for those came from Scrum. They are even used the same way in planning and performing iterations. (and – you can have your 30 day iterations too!) If you are a RUP project manager you will find that the process around the work items list and iteration plan is tangible and easy to manage.

OpenUP uses 'use cases' as envelope for requirements, but that doesn't preclude the use of user stories as a means to solicit feedback on the evolving system. Use cases and user stories are quite similar in nature, although use cases are coarser-grained than user stories are normally, and they come about in a more pro-active way.

As we have already seen, OpenUP acknowledges the value of architecture and regards it as an intentional property of the system. However, it doesn't treat it as a "large big up-front design" of the whole system. On the contrary, the activity of establishing the architecture is light-weight and focused on a small, central subset of the system's requirements and main constraints and objectives. This, too, is an iterative and incremental activity that occurs in each iteration, primarily during the inception and elaboration phases, in parallel with other project activities.

Of course, the approach to architecture taken by a particular project shall be determined by the circumstance of that project – the more complex and critical the project is, the more important becomes the architecture, and the more pro-active should the formulation of the architecture be.

The future of OpenUP

Like many other open source projects, OpenUP evolve in the direction set by the user community, so what the longer-term future has in stock for us is unclear.

We can rest assured though that, contrary to RUP and other proprietary processes, OpenUP will grow to include the practices most useful to the larger number of people in the software community.

Short-term, taking a quick peek behind the "development curtain", we see work currently being done to add the concept of 'practices' as primary building block, to replace the coarser-grained 'method plug-in' concept. This directly benefits all stakeholders of EPF and OpenUP, and ultimately serves you, the project member, with a more to-the-point process.

Other initiatives are integrating Wiki technology into EPF, to make augmenting and modifying process even easier.

Even if EPF is still a young initiative it has already started to draw the attention of tool vendors to create integrations with project management tools. This includes ProjectKoach [9], from GOOD Software Inc, which can import OpenUP activity models into its Eclipse-based iteration plans. Other tools offer similar capabilities, including Iris from Osellus [10]. Worth noting is that the future IBM Jazz [8] project incorporates EPF as one of its key components.

Summary

The new OpenUP process synthesizes the best practices from RUP and Agile methods into a light-weight and agile alternative to both. Given its RUP roots, it provides a

process that is iterative and incremental, use-case driven, risk-driven, and architecture-centric, which at the same time supports the work habits of agile projects.

The OpenUP process features practices suitable for many projects right out-of-the-box, but it also provides the basis for adding 3rd party and proprietary practice extensions, using the EPF Composer tool, to tailor the most appropriate process for each project.

If you haven't looked at OpenUP yet, I invite you to take a closer look at the EPF website [1] where you find more information and downloads.

References

[1] Eclipse Process Framework main web site www.eclipse.org/epf

[2] Agile Manifesto http://agilemanifesto.org/

[3] XP Extreme Programming http://www.extremeprogramming.org/

[4] Ken Schwaber and Mike Beedle *Agile Software Development with SCRUM*, Prentice Hall 2001

[5] SPEM specification http://www.omg.org/cgi-bin/doc?ptc/07-11-01

[6] Per Kroll *OpenUP in a nutshell*
http://www.ibm.com/developerworks/rational/library/sep07/kroll/index.html

[7] Dean Leffingwell *Scaling Software Agility*, Addison-Wesley 2007
http://www.leffingwell.org/ssa.html

[8] https://jazz.net

[9] ProjectKoach project management tool *www.projectkoach.com*
(GOOD Software Inc)

[10] Iris process enactment tools www.osellus.com

[11] Philippe Kruchten *The Rational Unified Process, an introduction* 2nd Ed. Addison-Wesley 2000