Introduction to OpenUP (Open Unified Process)

Different projects have different process needs. Typical factors dictate the needs for a more formal or agile process, such as team size and location, architecture complexity, technology novelty, conformance to standards, among others. Nevertheless, there are good software development practices that benefit any project team and help them to be more effective.

This paper introduces the building blocks of OpenUP – an agile and Unified Process that contains the minimal set of practices that help teams to be more effective in developing software. OpenUP embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic, low-ceremony process that can be used as is or extended to address a broad variety of project types.

What is OpenUP

OpenUP is a minimally sufficient software development process – meaning that only fundamental content is included. Thus, it does not provide guidance on many topics that projects may deal with, such as large team sizes, compliance, contractual situations, safety or mission critical applications, technology-specific guidance, etc. However, OpenUP is complete in the sense it can be manifested as an entire process to build a system. For addressing needs that are not covered in its content, OpenUP is extensible to be used as foundation on which process content can be added or tailored as needed.

OpenUP is an agile process. Though OpenUP is lightweight, there is much more to agility than simply being light. Most recognized agile practices are intended to get a team communicating with one another providing a shared understanding of the project. Agile methods have drawn our attention back to the importance of coordinating understanding, benefiting stakeholders over unproductive deliverables and formality.

OpenUP has the essential characteristics of a lean Unified Process that applies iterative and incremental approaches within a proven structured lifecycle. OpenUP is based on use cases and scenarios, risk management, and an architecture-centric approach to drive development.

OpenUP principles

OpenUP is driven by the four core principles listed below. Principles capture the general intentions behind a process and create the foundation for interpreting roles and work products, and for performing tasks:

- **Collaborate to align interests and share understanding**. This principle promotes practices that foster a healthy team environment, enable collaboration and develop a shared understanding of the project.
- **Balance competing priorities to maximize stakeholder value**. This principle promotes practices that allow project participants and stakeholders to develop a solution that maximizes stakeholder benefits, and is compliant with constraints placed on the project.
- **Focus on the architecture early to minimize risks and organize development**. This principle promotes practices that allow the team to focus on architecture to minimize risks and organize development.
- **Evolve to continuously obtain feedback and improve**. This principle promotes practices that allow the team to get early and continuous feedback from stakeholders, and demonstrate incremental value to them.

Each OpenUP principle supports a statement in the Agile Manifesto, as seen in Table 1.

OpenUP principle	**Agile Manifesto statement**
Collaborate to align interests and share understanding	Individuals and interactions over process and tools
Balance competing priorities to maximize stakeholder value	Customer collaboration over contract negotiation
Focus on the architecture early to minimize risks and organize development	Working software over comprehensive documentation
Evolve to continuously obtain feedback and improve	Responding to change over following a plan

Table 1 – Mapping between OpenUP principles and Agile Manifesto

How OpenUP is organized

OpenUP is organized in 2 different, correlated dimensions: method content and process content. The method content is where method elements (namely roles, tasks, artifacts, and guidance) are defined, regardless of how they are used in a project lifecycle. The process content is where the method elements are applied in a temporal sense. Many different lifecycles for different project types can be created from the same set of method elements (more details are on Process section below).

Content Areas

The OpenUP content addresses organization of work at personal, team and stakeholder levels, as seen in Figure 1.

At a personal level, team members in an OpenUP project contribute their work in **micro-increments**, which typically represent the outcome of a few hours to a few days of work. The application evolves one micro-increment at the time and progress is effectively seen every day. Team members openly share their daily progress on micro-increments, which increases work visibility, trust and teamwork.

The project is divided into iterations: planned, time-boxed intervals typically measured in weeks. OpenUP helps the team appropriately focus its effort through the **iteration lifecycle**, in order to deliver incremental value to stakeholders in a predictable manner - a fully tested demo-able or shippable build (product increment) at the end of each iteration.

OpenUP structures the **project lifecycle** into four phases: Inception, Elaboration, Construction, and Transition. The project lifecycle provides stakeholders with oversight, transparency, and steering mechanisms to control project funding, scope, risk exposure, value provided, and other aspects of the process.

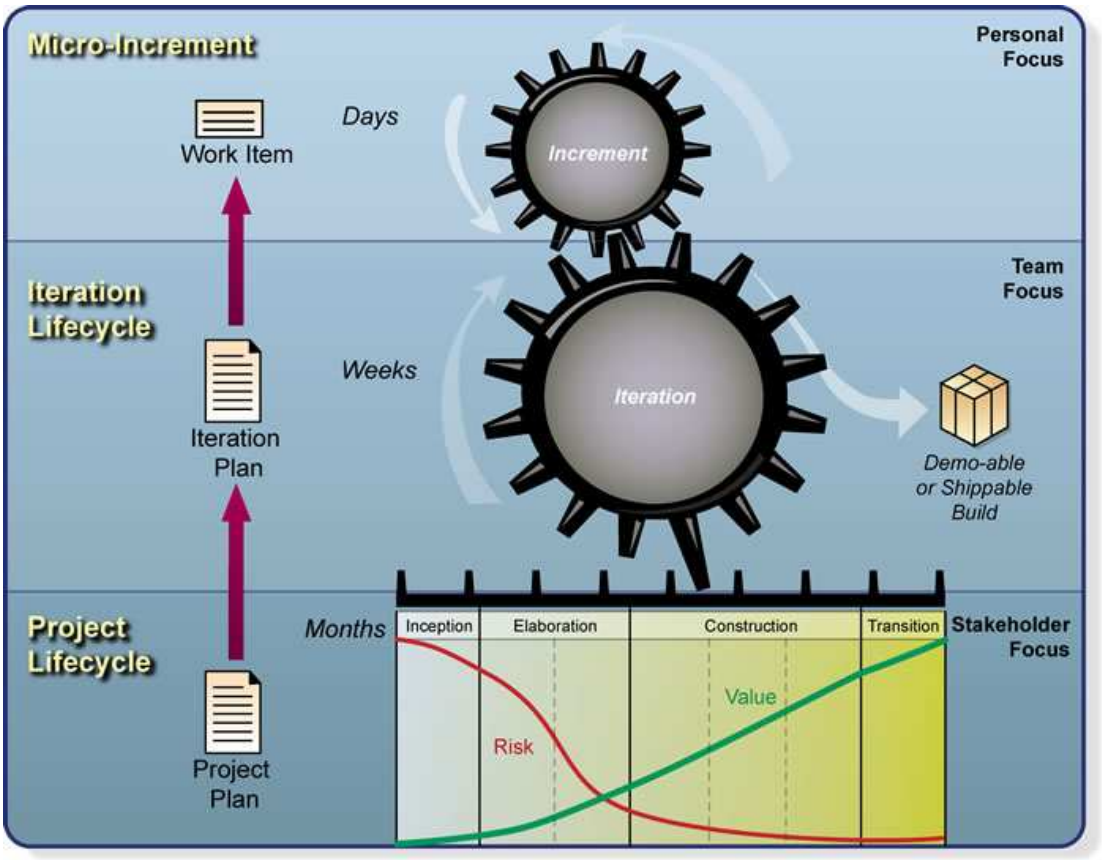

Figure 1 – Organization of work and content focus in OpenUP

Roles

The essential skills needed by small and co-located teams are represented by OpenUP roles:

- **Stakeholder** represents interest groups whose needs must be satisfied by the project. It is a role that may be played by anyone who is (or potentially will be) materially affected by the outcome of the project
- **Analyst** represents customer and end-user concerns by gathering input from stakeholders to understand the problem to be solved and by capturing and setting priorities for requirements.
- **Architect** is responsible for designing the software architecture, which includes making the key technical decisions that constrain the overall design and implementation of the project.
- **Developer** is responsible for developing a part of the system, including designing it to fit into the architecture, and then implementing, unit-testing, and integrating the components that are part of the solution.
- **Tester** is responsible for the core activities of the test effort, such as identifying, defining, implementing, and conducting the necessary tests, as well as logging the outcomes of the testing and analyzing the results.
- **Project Manager** leads the planning of the project in collaboration with stakeholders and team, coordinates interactions with the stakeholders, and keeps the project team focused on meeting the project objectives.
- **Any Role** represents anyone on the team that can perform general tasks.

Disciplines

The OpenUP method content is focused on the following disciplines: Requirements, Architecture, Development, Test, Project Management, and Configuration & Change Management.

Other disciplines and areas of concern were omitted, such as Business Modeling, Environment, advanced Requirements Management and Configuration Management tools setup. These concerns are either considered unnecessary for a small project or are handled by other areas of the organization, outside the project team.

Tasks

A task is unit of work a role may be asked to perform. In OpenUP, there are 18 tasks that the roles perform either as primary performers (the responsible for executing the task) or additional performers (supporting and providing information used in the task execution). The collaborative nature of OpenUP is manifested by having the primary performers work with a range of other individuals when performing a task.

Artifacts

An artifact is something that is produced, modified, or used by a task. Roles are responsible for creating and updating artifacts. Artifacts are subject to version control throughout the project lifecycle.

The 17 artifacts in OpenUP are considered the essential artifacts a project should use to capture product- and project-related information. There is no obligation in capturing information in formal artifacts. The information could be informally captured in whiteboard (e.g., for design and architecture), meeting notes (e.g., for status assessments), etc. Templates though provide an out-of-the box, standard way to capture information. Projects can use the OpenUP artifacts or replace them with their own.

Process

Reusable method content is created separately from its application in processes. Method content provides step-by-step explanations, describing how specific development goals are achieved independent of the placement of method elements within a development lifecycle.

Processes take these method elements and relate them into semi-ordered sequences that are customized to specific types of projects. Method elements are organized into reusable pieces of process called *capability patterns*, providing a consistent development approach to common project needs. These patterns are made from organizing tasks (from the method content) into activities, grouping them in a sequence that makes sense for the particular area where that pattern is applied.

Patterns can be small and focused in particular areas like iteration management, project initiation, architecture definition and so on. These are considered the basic building blocks to create larger patterns or delivery processes (defined below).

One example of basic building block in OpenUP is Develop Solution Increment pattern, as depicted in Figure 2.

This activity provides a way to perform goal-based planning and execution of work. Work is taken on by developers, and work progress is tracked based on the goals achieved using the designed, developer-tested, and integrated source code.

The work item can be a use case, a scenario, a supporting requirement or a change request. A context can be specified when a work item is assigned to be developed, thus specifying how broadly a work item is to be developed in that increment. Development may be focused, for example, on a layer (e.g., user-interface, business logic or database access), or on a component. Whether a context is specified or not, the developer's responsibility is to create a design and implementation for that work item, and to write and run developer tests against the implementation to make sure the implementation works as designed, both as a unit and integrated into the code base

Develop Solution Increment pattern occurs as many times as there are work items to be developed in a given iteration.

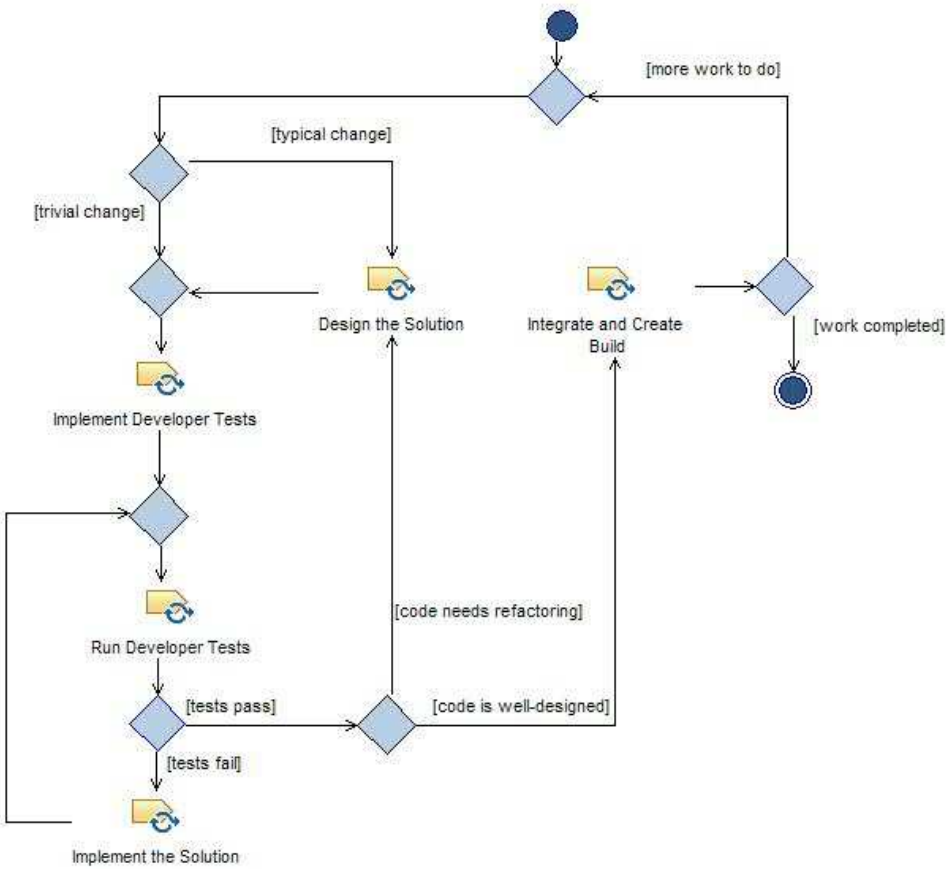

Figure 2 – Develop Solution Increment

As mentioned before, basic building blocks are used to create larger patterns, for example patterns that can be used as templates for iterations - patterns containing all activities needed for a particular iteration within a project phase.

OpenUP applies the Unified Process phases: Inception, Elaboration, Construction, and Transition. When taken together, these basic building blocks are also used to address the objectives for each phase (see Table 2 for a mapping between patterns and phases objectives).

Iteration template patterns	Phase objectives
Inception Phase Iteration Initiate Project Plan and Manage Iteration Identify and Refine Requirements Agree on Technical Approach	▪ Understand what to build ▪ Identify key system functionality ▪ Determine at least one possible solution ▪ Understand the cost, schedule and risks associated with the project
Elaboration Phase Iteration Plan and Manage Iteration Identify and Refine Requirements Define the Architecture Develop Solution Increment Test Solution Ongoing Tasks	▪ Get a more detailed understanding of the requirements ▪ Design, implement, validate, and baseline an Architecture ▪ Mitigate essential risks, and produce accurate schedule and cost estimates
Construction Phase Iteration Plan and Manage Iteration Identify and Refine Requirements Develop Solution Increment Test Solution Ongoing Tasks	▪ Iteratively develop a complete product that is ready to transition to its user community ▪ Minimize development costs and achieve some degree of parallelism
Transition Phase Iteration Plan and Manage Iteration Develop Solution Increment Test Solution Ongoing Tasks	▪ Beta test to validate that user expectations are met ▪ Achieve stakeholder concurrence that deployment is complete

Table 2 – mapping between patterns and phases objectives

When we sequence the iteration template patterns (occurring as many times as needed), we have a *delivery process* – a complete and integrated approach for performing a specific project type (as seen in Figure 3). A delivery process describes a complete project lifecycle and is used as a reference for running similar projects.

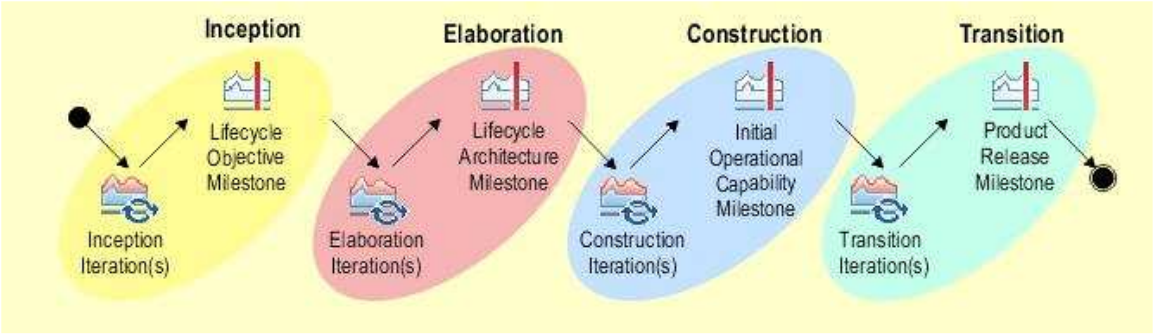

Figure 3 – OpenUP delivery process

OpenUP has a delivery process for iterative development throughout four phases. The iteration template patterns are applied as many times as needed, depending on how many iterations the team decide to run in each phase. Projects with different needs may instantiate iteration template patterns differently. For example, a project dealing with an unknown technology or complex architecture may need more iterations in Elaboration phase than a project dealing with a known technology or pre-existing architecture.

A base process

OpenUP process, though complete in its target coverage and context, also serves as a base process upon which additional process content can be built. Plug-ins can extend OpenUP to include guidance for large-scale techniques (such as Model-Driven Development) and lighter, agile techniques (such as Agile Database Techniques). Tool vendors can create plug-ins that attach tool mentors to tasks giving step-by-step instructions on how the tool can be used within the context of the process.

In addition to this community of plug-ins that can be mixed and matched to build a tailored process – and possibly more important for the goal of having a process that meets the specific needs of a particular project – the Eclipse Process Framework Composer tool can be used to author in-house content. Organization-specific templates can be integrated into the process content and new process elements from checklists and guidance to brand new roles, tasks, and work products can be introduced.

Conclusion

OpenUP is a minimal, complete and extensible process. It fosters agile techniques and principles, while has a proven structured lifecycle that emphasizes the continuous delivery of quality software that is valuable to stakeholders.

Acknowledgements

This article expresses ideas and borrows content from OpenUP, which is the result of an open-source project accomplished by the cooperation of various individuals and companies.

References

- Agile Manifesto: www.agilemanifesto.org
- Eclipse Process Framework project: www.eclipse.org/epf
- Lyons, B.: The Open Unified Process: A Brilliant, Collaborative March into Open Source - http://www.numbersix.com/news/n6articles/openUp.html

About the author

Ricardo Balduino is senior software engineer at IBM Rational and a committer on the Eclipse Process Framework project. He is also content architect and content developer lead for OpenUP. His 13 years of experience in the software industry includes developing software, delivering training and consulting services to help organizations to customize and adopt software development best practices. He holds a Bachelor of Science degree in Computer Science from São Paulo State University, Brazil.

August, 2007.