

Basic Unified Process: A Process for Small and Agile Projects

Ricardo Balduino - Rational Unified Process Content Developer, IBM

Introduction

Small projects have different process needs than larger projects. Smaller project teams generally want low overhead, so they can focus on delivering the product. Nevertheless, there are good practices that benefit small projects, and help them to be more effective.

This paper introduces the Basic Unified Process – a process focused on practices suited to most small projects and teams.

What is the Basic Unified Process

Basic Unified Process (BUP) is a streamlined version of IBM Rational Unified Process (RUP) optimized for small projects. Small projects constitute teams of 3 to 6 people and involve 3 to 6 months of development effort. BUP preserves the essential characteristics of RUP, which includes iterative development, use-cases and scenarios driving development, risk management, and architecture-centric approach. Most optional parts of RUP have been excluded, and many elements have been merged. The result is a much simpler process that is still true to RUP principles.

How the Basic Unified Process is organized

BUP is organized in 2 different (but correlated) dimensions: method and process. The method dimension is where method elements (namely roles, tasks, artifacts, and guidance) are defined, regardless of how they are applied on a project lifecycle. The process dimension is where the method elements are applied in a behavioral sense, where from the same set of method elements, many different lifecycles for different types of projects can be created (more details on section **Process** ahead).

Disciplines

BUP method content is focused on the following disciplines: requirements, architecture, development, test, project management and change management. Content around these various disciplines are organized into packages, allowing selection of only desired content when creating a configuration for publishing.

Other disciplines and areas of concern were omitted, like business modeling, environment, advanced requirements management, configuration management, etc., for those concerns are considered advanced for a small project or are handled by other areas of the organization, not by the project team.

Analysis & design content is not called out in a separate discipline, but it was absorbed by other disciplines like architecture and development. The content was absorbed because the architect does some high level analysis when he/she identifies key abstractions, high level layering, reuse of existing solutions and patterns, and so on. High level design is also done by architect when major components and their interfaces are identified.

On the other hand, developer performs low level analysis and design, by identifying classes and internal parts of components. Development discipline concentrates tasks the developer performs to evolve the design into implementation, which is unit-tested and integrated into the code base.

Big up-front design is not emphasized in BUP. Design can happen either before or at the same time the developer writes code and unit tests. The emphasis is more on thinking the design than capturing it in visual models. We expect that projects can extend BUP to add variations on how to capture design decisions, like CRC cards, UML diagrams on whiteboard, and so on.

Roles

The essential skills needed by small and co-located teams are represented by BUP roles.

- **Analyst** – responsible for gathering requirements and documenting them as needed. In small agile projects, a customer representative may play this role.
- **Architect** – responsible for the software architecture, which includes the key technical decisions that constrain the overall design and implementation for the project.
- **Developer** – create a solution (or part of it) by doing design, implementation, unit tests and integration of components.
- **Tester** – responsible for testing the system from a larger perspective than the developer does, making sure the system works as defined and is accepted by the customer.
- **Project Manager** – plans and manages the project, coordinates interactions with the stakeholders, and keeps the project team focused.
- **Any Role** represents anyone in the team who can perform tasks like submitting and performing changes, participating in meetings, reviewing, etc.

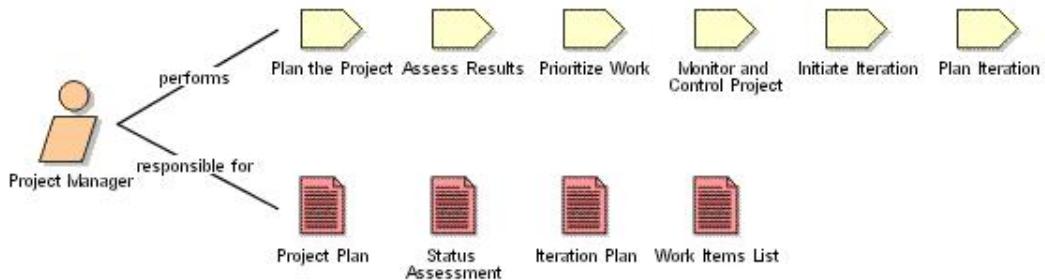
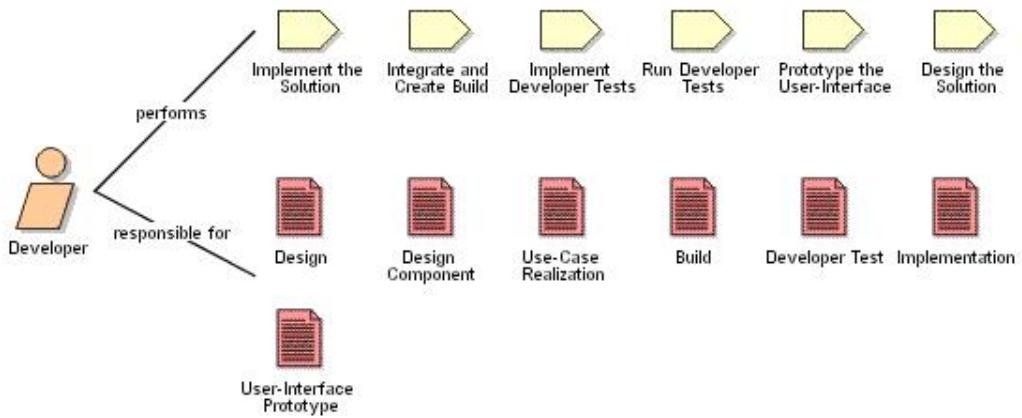
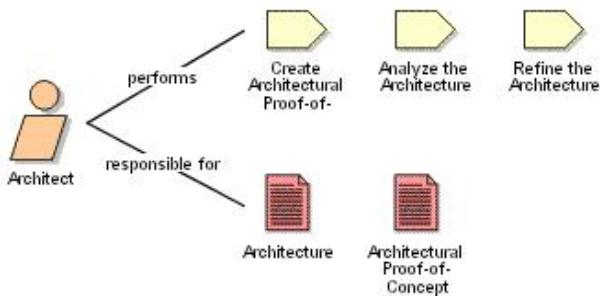
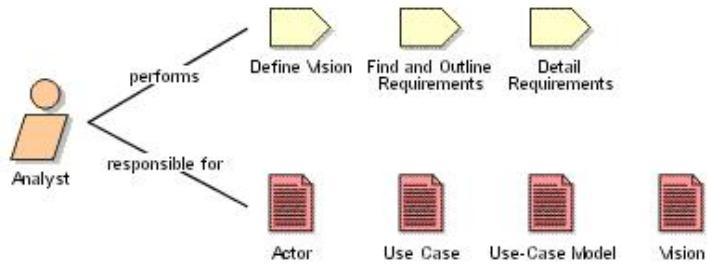
Tasks

Only low-level-of-ceremony tasks were considered in BUP, those typically needed by small projects. Some RUP tasks were transformed into guidelines, being referenced by simpler tasks. Example is “Find and Outline Actors and Use Cases”.

Some RUP tasks were transformed into steps and included inside another major task being performed by the same role at the same point in time. A project may decide if that step is performed or not, depending on what is needed. Examples: “Handle Exception and Problems” and “Identify and Manage Risks” became steps in “Monitor and Control Project” task.

Artifacts

Low level of ceremony is obtained not only by the reduced number of artifacts (compared to RUP), but also because only 6 of these artifacts actually have (informal) templates. The remaining artifacts have guidelines explaining how to informally represent them. In general, these guidelines recommend capturing the information in an existing artifact, spreadsheet, database, table, e-mail, etc. This allows projects to select the appropriate and lowest level of formality required for artifacts.



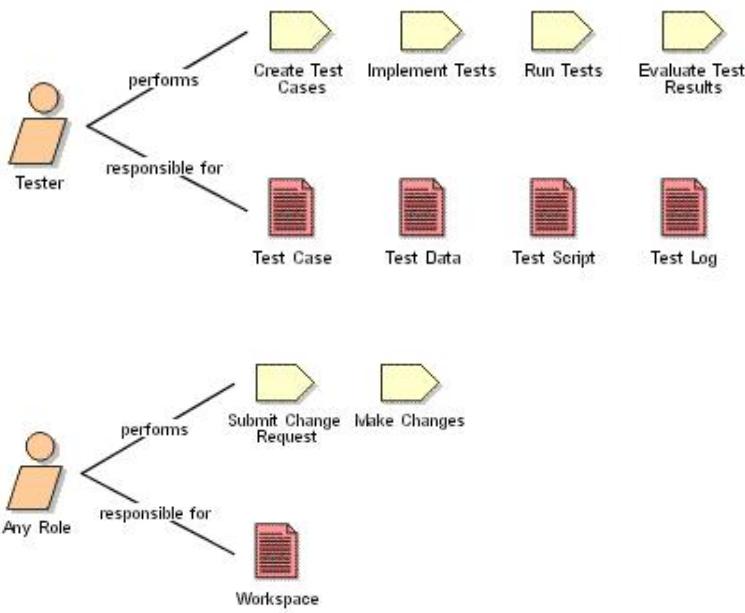


Figure 1 – list of roles, tasks and artifacts in BUP

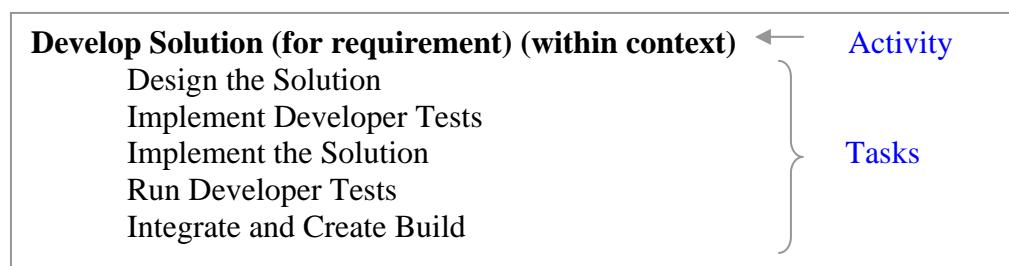
Process

Reusable method content is created separate from its application in processes. Method content provides step-by-step explanations, describing how specific development goals are achieved independent of the placement of these steps 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 problems. These patterns are made of activities organizing tasks (from the method content), 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 BUP is *Develop Solution*, as depicted below:



Project managers use this pattern as a way to perform goal-based planning and status reporting. Work is assigned to developers and work progress is tracked based on the goals to be achieved,

i.e., the designed, implemented, unit-tested and integrated source code. The assigned requirement can be a use case, a scenario or a supplemental requirement. A context can be specified when a requirement is assigned to be developed, thus specifying how broadly a requirement is to be developed by that person – development may be focused on a layer (e.g., user-interface, business logic or database access), on a component and so on. Whether a context is specified or not, developer's responsibility is to create a design and implementation for that requirement, then to write and run unit tests against the implementation to make sure the implementation works as designed, both as a unit and integrated into the code base. This pattern occurs as many times as there are requirements to be developed in a given iteration.

As mentioned, basic building blocks are used to create larger patterns, for example, *iteration template patterns* – patterns containing all activities needed for a particular iteration within a project phase. When taken together, these basic building blocks not only are used to assemble iteration template patterns, but when performed they also address the objectives for each phase (see Table 1 for a mapping between patterns and phases objectives).

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

Iteration template patterns	Phase objectives
 Inception Phase Iteration <ul style="list-style-type: none"> <li data-bbox="298 977 556 1009"> Manage Iteration <li data-bbox="298 1009 523 1041"> Initiate Project <li data-bbox="298 1041 616 1072"> Manage Requirements <li data-bbox="298 1072 780 1104"> Determine Architectural Feasibility <li data-bbox="298 1104 714 1136"> Assess and Plan Next Iteration 	<ul style="list-style-type: none"> <li data-bbox="967 952 1432 984">▪ Understand what to build <li data-bbox="967 984 1432 1015">▪ Identify key system functionality <li data-bbox="967 1015 1432 1089">▪ Determine at least one possible solution <li data-bbox="967 1089 1432 1142">▪ Understand the cost, schedule and risks associated with the project
 Elaboration Phase Iteration <ul style="list-style-type: none"> <li data-bbox="298 1199 556 1231"> Manage Iteration <li data-bbox="298 1231 616 1262"> Manage Requirements <li data-bbox="298 1262 633 1294"> Define the Architecture <li data-bbox="298 1294 861 1368"> Develop Solution (for requirement) (within context) <li data-bbox="298 1368 518 1400"> Validate Build <li data-bbox="298 1400 714 1431"> Assess and Plan Next Iteration <li data-bbox="298 1431 556 1463"> Manage Changes 	<ul style="list-style-type: none"> <li data-bbox="967 1174 1441 1227">▪ Get a more detailed understanding of the requirements <li data-bbox="967 1227 1441 1300">▪ Design, implement, validate, and baseline an Architecture <li data-bbox="967 1300 1441 1417">▪ Mitigate essential risks, and produce accurate schedule and cost estimates
 Construction Phase Iteration <ul style="list-style-type: none"> <li data-bbox="298 1516 556 1548"> Manage Iteration <li data-bbox="298 1548 616 1579"> Manage Requirements <li data-bbox="298 1579 633 1611"> Refine the Architecture <li data-bbox="298 1611 861 1685"> Develop Solution (for requirement) (within context) <li data-bbox="298 1685 518 1717"> Validate Build <li data-bbox="298 1717 714 1748"> Assess and Plan Next Iteration <li data-bbox="298 1748 556 1780"> Manage Changes 	<ul style="list-style-type: none"> <li data-bbox="967 1491 1454 1586">▪ Iteratively develop a complete product that is ready to transition to its user community <li data-bbox="967 1586 1437 1638">▪ Minimize development costs and achieve some degree of parallelism

 Transition Phase Iteration  Manage Iteration  Develop Solution (for requirement) (within context)  Validated Build  Assess and Plan Next Iteration  Manage Changes	<ul style="list-style-type: none"> ▪ Beta test to validate that user expectations are met ▪ Achieve stakeholder concurrence that deployment is complete
--	---

Table 1 – mapping between patterns and phases objectives

BUP has a delivery process for iterative development throughout four phases. The iteration template patterns are put together, as many times as needed, depending on how the project manager needs to instantiate them to create a project plan, as exemplified in Figure 2.

Projects with different needs may need to instantiate iteration template patterns differently, for example a project dealing with an unknown technology or architecture may want to perform 3 iterations in Elaboration.

Either each individual iteration template pattern or the whole delivery process can be used to instantiate a project plan, or parts of a project plan. The reason is that capability patterns and delivery process contain Work Breakdown Structures (WBS) with tasks organized under activities. The WBS can be used to populate a project plan, where the project manager assigns resources to perform tasks. This bridges the gap between process management and project management, since project plans reflect the exact set of activities and tasks defined for a given delivery process. To avoid creating overly detailed plans, the project manager may create the plan incrementally, and using only higher level activities, leveraging lower level tasks only as a guide for how to do the work.

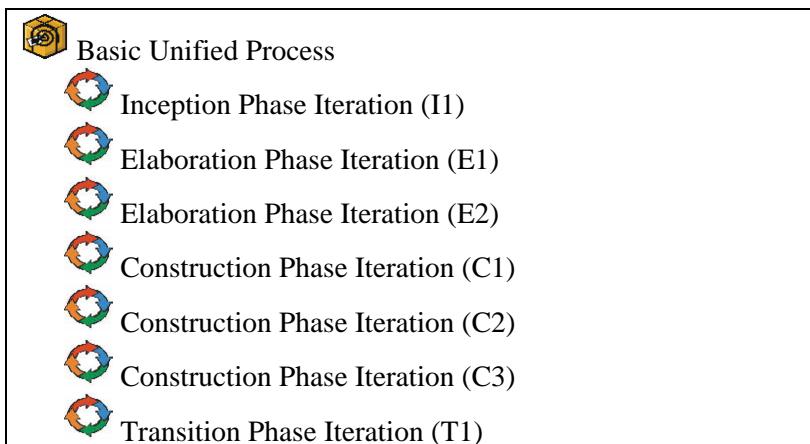


Figure 2 – example of delivery process

Conclusion

BUP supports agility in the sense it has a small number of roles, tasks and informal artifacts. It supports flexibility, since its organization in packages and separation between method and

process allows selection of desired method elements (only a subset, if one will) to create the process that make sense to the project reality.

About the author

Ricardo Balduino is a senior software engineer and IBM Rational Unified Process,® or RUP,® content developer at IBM. Previously, he spent four years as a software engineering specialist at IBM Rational in Brazil, delivering training and consulting services to customers in various industries and helping organizations to customize and adopt RUP. He has also developed software for financial services, industrial process automation, and other specialties. He holds a B.S. in computer science from the University of Sao Paulo State, Brazil.