Rational Unified Process

From Wikipedia, the free encyclopedia

The **Rational Unified Process** (**RUP**) is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003.[1] RUP is not a single concrete prescriptive process, but rather an adaptable process framework, intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs. RUP is a specific implementation of the Unified Process.

Contents

History

Rational Software originally developed the rational unified process as a software process product. The product includes a hyperlinked knowledge-base with sample artifacts and detailed descriptions for many different types of activities. RUP is included in the IBM Rational Method Composer (RMC) product which allows customization of the process.

Philippe Kruchten, an experienced Rational technical representative was tasked with heading up the original RUP team. This journey began with the creation of the Rational Objectory Process (ROP) in 1996, when Rational acquired the Objectory Process that had been written by Ivar Jacobson and company. This was renamed Rational Unified Process (RUP) in subsequent releases, in part to align the name with that of the Unified Modeling Language.

These initial versions combined the Rational Software organisation's extensive field experience building object-oriented systems (referred to by Rational field staff as the Rational Approach) with Objectory's guidance on practices such as use cases, and incorporated extensive content from Jim Rumbaugh's Object Modeling Technology (OMT) approach to modeling, Grady Booch's Booch method, and the newly released UML 0.8.[2][3]

To help make this growing knowledge base more accessible, Philippe Kruchten was tasked with the assembly of an explicit process framework for modern software engineering. This effort employed the HTML-based process delivery mechanism developed by Objectory. The resulting "Rational Unified Process" (RUP) completed a strategic tripod for Rational:

- a *tailorable process* that guided development
- *tools* that automated the application of that process
- *services* that accelerated adoption of both the process and the tools.

This guidance was augmented in subsequent versions with knowledge based on the experience of companies that Rational had acquired.

In 1997, a requirements and test discipline were added to the approach, much of the additional material sourced from the Requirements College method developed by Dean Leffingwell et al. at Requisite, Inc., and the SQA Process method developed at SQA Inc., both companies having been acquired by Rational Software.

In 1998 Rational Software added two new disciplines:

1. business modeling, much of this content had already been in the Objectory Process
2. a Configuration and Change Management discipline, sourced through the acquisition of Pure Atria Corporation.

These additions lead to an overarching set of principles that were defined by Rational and articulated within RUP as the six *best practices* for modern software engineering:

1. Develop iteratively, with risk as the primary iteration driver[4]
2. Manage requirements
3. Employ a component-based architecture
4. Model software visually
5. Continuously verify quality
6. Control changes

These best practices were tightly aligned with Rational's product line, and both drove the ongoing development of Rational's products, as well as being used by Rational's field teams to help customers improve the quality and predictability of their software development efforts.

Additional techniques including performance testing, UI Design, data engineering were included, and an update to reflect changes in UML 1.1.

In 1999, a project management discipline was introduced, as well as techniques to support real-time software development and updates to reflect UML 1.3

Between 2000 and 2003, a number of changes introduced guidance from ongoing Rational field experience with iterative development, in addition to tool support for enacting RUP instances and for customization of the RUP framework. These changes included:

1. the introduction of concepts and techniques from approaches such as eXtreme Programming (XP), that would later come to be known collectively as agile methods. This included techniques such as pair programming, test-first design, and papers that explained how RUP enabled XP to scale for use on larger projects.
2. a complete overhaul of the testing discipline to better reflect how testing work was conducted in different iterative development contexts.
3. the introduction of supporting guidance - known as "tool mentors" - for enacting the RUP practices in various tools. These essentially provided step-by-step method support to Rational tool users.
4. automating the customization of RUP in a way that would allow customers to select parts from the RUP process framework, customize their selection with their own additions, and still incorporate improvements in subsequent releases from Rational.

IBM acquired Rational Software in February 2003.

In 2006, IBM created a subset of RUP tailored for the delivery of Agile projects - released as an OpenSource method called OpenUP through the Eclipse web-site.[5]

Rational unified process topics

RUP building blocks

RUP is based on a set building blocks and content elements, describing what is to be produced, the necessary skills required and the step-by-step explanation describing how specific development goals are to be achieved. The main building blocks, or content elements, are the following:

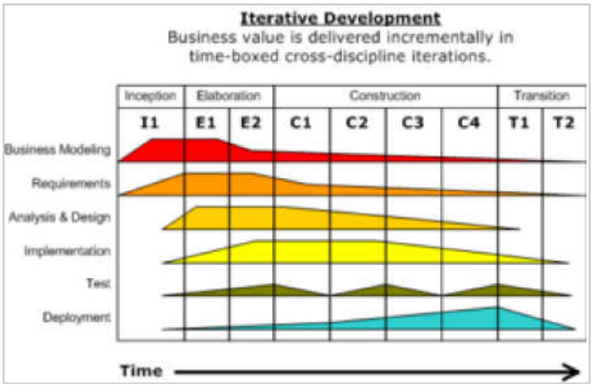
- Roles (who) – A role defines a set of related skills, competencies and responsibilities.
- Work products (what) – A work product represents something resulting from a task, including all the documents and models produced while working through the process.
- Tasks (how) – A task describes a unit of work assigned to a Role that provides a meaningful result.

Within each iteration, the tasks are categorized into nine disciplines:

- Six "engineering disciplines"
    - Business modelling
    - Requirements
    - Analysis and design
    - Implementation
    - Test
    - Deployment
- Three supporting disciplines
    - Configuration and change management
    - Project management
    - Environment

Four project life-cycle phases

The RUP has determined a project life-cycle consisting of four phases. These phases allow the process to be presented at a high level in a similar way to how a 'waterfall'-styled project might be presented, although in essence the key to the process lies in the iterations of development that lie within all of the phases. Also, each phase has one key objective and milestone at the end that denotes the objective being accomplished. The visualization of RUP phases and disciplines over time is referred to as the RUP hump chart.



RUP phases and disciplines.

Inception phase

The primary objective is to scope the system adequately as a basis for validating initial costing and budgets. In this phase the business case which includes business context, success factors (expected revenue, market recognition, etc.), and financial forecast is established. To complement the business case, a basic use case model, project plan, initial risk assessment and project description (the core project requirements, constraints and key features) are generated. After these are completed, the project is checked against the following criteria:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Establishing a baseline by which to compare actual expenditures versus planned expenditures.

If the project does not pass this milestone, called the life cycle objective milestone, it either can be cancelled or repeated after being redesigned to better meet the criteria.

Elaboration phase (Ortner)

The primary objective is to mitigate the key risk items identified by analysis up to the end of this phase. The elaboration phase is where the project starts to take shape. In this phase the problem domain analysis is made and the architecture of the project gets its basic form.

The outcome of the elaboration phase is:

- A use-case model in which the use-cases and the actors have been identified and most of the use-case descriptions are developed. The use-case model should be 80% complete.
- A description of the software architecture in a software system development process.
- An executable architecture that realizes architecturally significant use cases.
- Business case and risk list which are revised.
- A development plan for the overall project.
- Prototypes that demonstrably mitigate each identified technical risk.
- A preliminary user manual (optional)

This phase must pass the lifecycle architecture milestone criteria answering the following questions:

- Is the vision of the product stable?
- Is the architecture stable?
- Does the executable demonstration indicate that major risk elements are addressed and resolved?
- Is the construction phase plan sufficiently detailed and accurate?
- Do all stakeholders agree that the current vision can be achieved using current plan in the context of the current architecture?
- Is the actual vs. planned resource expenditure acceptable?

If the project cannot pass this milestone, there is still time for it to be canceled or redesigned. However, after leaving this phase, the project transitions into a high-risk operation where changes are much more difficult and detrimental when made.

The key domain analysis for the elaboration is the system architecture.

Construction phase

The primary objective is to build the software system. In this phase, the main focus is on the development of components and other features of the system. This is the phase when the bulk of the coding takes place. In larger projects, several construction iterations may be developed in an effort to divide the use cases into manageable segments that produce demonstrable prototypes.

This phase produces the first external release of the software. Its conclusion is marked by the initial operational capability milestone.

Transition phase

The primary objective is to 'transit' the system from development into production, making it available to and understood by the end user. The activities of this phase include training the end users and maintainers and beta testing the system to validate it against the end users' expectations. The system also goes through an evaluation phase, any developer which is not producing the required work is replaced or removed. The product is also checked against the quality level set in the Inception phase.

If all objectives are met, the product release milestone is reached and the development cycle is finished.

The IBM Rational Method Composer product

The IBM Rational Method Composer product is a tool for authoring, configuring, viewing, and publishing processes. See IBM Rational Method Composer and an open source version Eclipse Process Framework (EPF) project for more details.

Certification

In January 2007 the new RUP certification examination for *IBM Certified Solution Designer - Rational Unified Process 7.0* was released which replaces the previous version of the course called *IBM Rational Certified Specialist - Rational Unified Process*.[6] The new examination will not only test knowledge related to the RUP content but also to the process structure elements.[7]

To pass the new RUP certification examination, a person must take IBM's *Test 839: Rational Unified Process v7.0*. You are given 75 minutes to take the 52 question exam. The passing score is 62%.[8]

Six best practices

Six best practices as described in the rational unified process is a paradigm in software engineering that lists six ideas to follow when designing any software project to minimize faults and increase productivity. These practices are:[9][10]

Develop iteratively
: It is best to know all requirements in advance; however, often this is not the case. Several software development processes exist that deal with providing solution on how to minimize cost in terms of development phases.

Manage requirements
: Always keep in mind the requirements set by users.

Use components
: Breaking down an advanced project is not only suggested but in fact unavoidable. This promotes ability to test individual components before they are integrated into a larger system. Also, code reuse is a big plus and can be accomplished more easily through the use of object-oriented programming.

Model visually
: Use diagrams to represent all major components, users, and their interaction. "UML", short for Unified Modeling Language, is one tool that can be used to make this task more feasible.

Verify quality
: Always make testing a major part of the project at any point of time. Testing becomes heavier as the project progresses but should be a constant factor in any software product creation.

Control changes
: Many projects are created by many teams, sometimes in various locations, different platforms may be used, etc. As a result, it is essential to make sure that changes made to a system are synchronized and verified constantly. (See Continuous integration).

See also

- Macroscope (methodology suite)
- Agile Modeling (AM)
- Agile Unified Process (AUP)
- Disciplined agile delivery (DAD)
- Dynamic Systems Development Method (DSDM)
- Computer programming
- Feature-driven development (FDD)
- Project life cycle
- Quality assurance

- Scaled Agile Framework — a descendent of RUP that incorporates Agile software development methods such as Extreme programming (XP)
- Software architecture
- Software component
- Software development process
- Software engineering
- Software testing
- Test-driven development (TDD)

References

1. IBM Acquires Rational (http://www.eweek.com/c/a/Desktops-and-Notebooks/IBM-Acquires-Rational/)
2. Jacobson, Sten (2002-07-19). "The Rational Objectory Process - A UML-based Software Engineering Process". Rational Software Scandinavia AB. Retrieved 2014-12-17.
3. Kruchten, Philippe (2004-05-01). "The Rational Unified Process: An Introduction". Addison-Wesley. p. 33. Retrieved 2014-12-17.
4. Aked, Mark (2003-11-25). "RUP in brief". IBM. Retrieved 2011-07-12.
5. http://epf.eclipse.org/wikis/openup/
6. Krebs, Jochen (2007-01-15). "The value of RUP certification". IBM. Retrieved 2014-05-05.
7. "Spacer IBM Certified Solution Designer - IBM Rational Unified Process V7.0". IBM. Retrieved 2008-05-13.
8. "Test 839: Rational Unified Process v7.0". IBM. Retrieved 2008-05-13.
9. Stephen Schach (2004). *Classical and Object-Oriented Software Engineering*. 6/e, WCB McGraw Hill, New York, 2004.
10. Rational Unified Process white paper (http://www.augustana.ab.ca/~mohrj/courses/2000.winter/csc220/papers/rup_best_practices/rup_bestpractices.html)

Further reading

- Ivar Jacobson, Grady Booch, and James Rumbaugh (1999). *The Unified Software Development Process*
- Gary Pollice, Liz Augustine, Chris Lowe, and Jas Madhur (2003). *Software Development for Small Teams: A RUP-Centric Approach*
- Per Kroll, Philippe Kruchten (2003). *Rational Unified Process Made Easy, The: A Practitioner's Guide to the RUP*
- Per Kroll, Bruce Mac Isaac (2006). *Agility and Discipline Made Easy: Practices from OpenUP and RUP*
- Philippe Kruchten (1998). *The Rational Unified Process: An Introduction*
- Ahmad Shuja, Jochen Krebs (2007). *RUP Reference and Certification Guide*
- Walker Royce, *Software Project Management, A Unified Framework*

External links

- IBM Rational Unified Process Web Site (http://www-306.ibm.com/software/awdtools/rup/)
- Rational Software at IBM (http://www.rational.com/)
- Global Rational User Group Community (http://www.rational-ug.org/)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Rational_Unified_Process&oldid=761315876"

Categories: Formal methods | IBM software | Software project management