IBM Rational Unified Process

From Wikipedia, the free encyclopedia
(Redirected from Rational Unified Process)

The **Rational Unified Process** (**RUP**) is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2002.

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.

Software development process

Activities and steps

Requirements | Architecture | Implementation | Testing | Deployment

Models

Agile | Cleanroom | Iterative | RAD | RUP | Spiral | Waterfall | XP

Supporting disciplines

Configuration management | Documentation | Project management | User experience design

The Rational Unified Process is also a software process product, originally developed by Rational Software, and now available from IBM. 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.

The Unified Process was designed from the start to include both a generic, public domain process (known as the Unified Process), and a more detailed specification known as the *Rational Unified Process* which could be marketed as a commercial product.

Contents

Background

History

The roots of Rational Process go back to the original spiral model of Barry Boehm. The Rational Approach was developed at Rational Software in the 1980s and 1990s.

In 1995 Rational Software acquired the Swedish Company Objectory AB. The Rational Unified Process was the result of the merger of the Rational Approach and the Objectory process developed by Objectory founder Ivar Jacobson. The first results of that merger was the Rational Objectory Process, designed to an Objectory-like process, but suitable to wean Objectory users to the Rational Rose tool. When that goal was accomplished, the name was changed. The first version of the Rational Unified Process, version 5.0, was released in 1998. The chief architect was Philippe Kruchten. The latest version of RUP (7.0) was released with the announcement [1] (http://news.thomasnet.com/fullstory/471030/2585) of the IBM Rational Method Composer in November 2005.

Design

The creators and developers of the process focused on diagnosing the characteristics of different failed software projects; by doing so they tried to recognize the root causes of these failures. They also looked at the existing software engineering processes and their solutions for these symptoms.

A representative list of failure causes includes the following:

- *Ad hoc* requirements management
- Ambiguous and imprecise communication
- Brittle architecture (architecture that does not work properly under "stress")
- Overwhelming complexity
- Undetected inconsistencies in requirements, designs, and implementations
- Insufficient testing
- Subjective assessment of project status
- Failure to attack risks
- Uncontrolled change propagation
- Insufficient automation

Project failure is caused by a combination of several symptoms, though each project fails in a unique way. The outcome of their study was a system of software best practices they named the Rational Unified Process.

The Process was designed with the same techniques the team used to design software; it has an underlying object-oriented model, using Unified Modeling Language (UML).

6 Key Principles of Business-Driven Development

RUP is based on a set of six key principles for business-driven development (http://www-128.ibm.com/developerworks/rational/library/oct05/kroll/) :

1. Adapt the process
2. Balance stakeholder priorities
3. Collaborate across teams
4. Demonstrate value iteratively
5. Elevate the level of abstraction

6. Focus continuously on quality

Adapt the process

A project or organization must right-size the process to their needs. For example, governance, size of the project, regulations etc, drive the degree of formality used in a project. RUP provides pre-configured process templates for small, medium and large projects, which can be used for easier adoption. The ceremony of the process should reflect the goals of the RUP phases. Adapting a process also encourages the continuous improvement of a process in an organization.

Balance stakeholder priorities

This key principle widens the discussion from a pure software requirements point of view, to a higher discussion. This includes business goals and stakeholder needs. Often, they compete and conflict, which needs to be balanced out between the parties involved.

Collaborate across teams

Software engineering is a team effort. With a broad variety of stakeholders, all voices need to be heard. With the increasing demand of globally distributed development, collaboration is enabled through modern communication tools. The collaboration is not limited to requirements, but includes exchange of metrics, test results, release management and project plans. That is especially true for RUP projects which are executed in an iterative-incremental approach.

Demonstrate value iteratively

Projects are delivered, even though often only internally, in increments in an iterative fashion. The increment, which includes the value of the past iteration, is used to measure the progress of the project. That increment is also used to encourage feedback from stakeholders about the direction of the project. This allows projects to adjust to changed situations based on the feedback. The stakeholders on the other side, can influence the shape of the development effort while the project is executed. The combination of the iterative development and the focus on risks in RUP, allows projects an iterative risk-assessment.

Elevate the level of abstraction

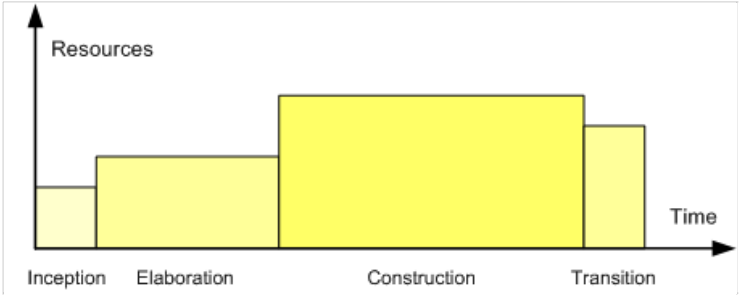
This key principle motivates the use of reusable assets such as software pattern, 4GL or Framework to name a few. This prevents software engineers going directly from the requirements to custom-made software code. A higher level of abstraction also allows discussions on different architectural levels. These can be accompanied by visual representations of the architecture, for example using UML.

Focus continuously on quality

Quality checks are not only at the end of each iteration but is a continuous ongoing activity in the software engineering project, often performed in a daily rhythm supported by the entire team. Automating test scenarios (scripts) helps dealing with the increasing amount of tests due to iterative development.UML

Project lifecycle

The RUP lifecycle is an implementation of the spiral model. It has been created by assembling the content elements into semi-ordered sequences. Consequently the RUP lifecycle is available as a work breakdown structure, which could be customized to address the specific needs of a project. The RUP lifecycle organizes the tasks into phases and iterations.



A typical project profile showing the relative sizes of the four phases

A project has four phases:

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Inception phase

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.
- Actual expenditures versus planned expenditures. ??

If the project does not pass this milestone, called the **Lifecycle Objective Milestone**, it can either be cancelled or it can repeat this phase after being redesigned to better meet the criteria.

Elaboration 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.

This phase must pass the **Lifecycle Architecture Milestone** by meeting the following criteria:

- 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 demonstratively mitigate each identified technical risk.

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

Construction phase

In this phase, the main focus goes to the development of components and other features of the system being designed. 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

In the transition phase, the product has moved from the development organization to the end user. The activities of this phase include training of the end users and maintainers and beta testing of the **system to validate it** against the end users' expectations. The product is also checked against the quality level set in the Inception phase. If it does not meet this level, or the standards of the end users, the entire cycle in this phase begins again.

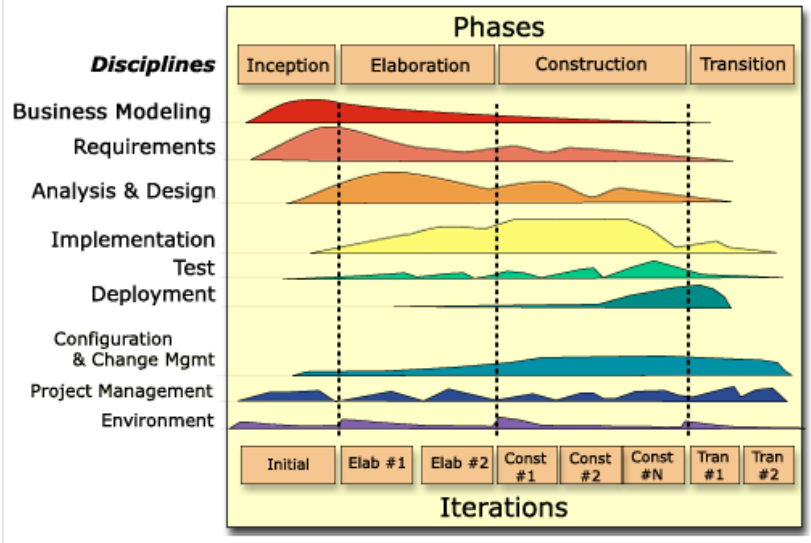
If all objectives are met, the **Product Release Milestone** is reached and the development cycle ends.

Disciplines and workflows

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

The main building blocks, or content elements, are the following:

- Roles (who) – A Role defines a set of related skills, competences, 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.



The Rational Unified Process

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

Engineering Disciplines:

- Business modeling discipline
- Requirements discipline
- Analysis and design discipline
- Implementation discipline
- Test discipline
- Deployment discipline

Supporting Disciplines:

- Configuration and change management discipline
- Project management discipline
- Environment discipline

Business modeling discipline

Organizations are becoming more dependent on IT systems, making it imperative that information system engineers know how the applications they are developing fit into the organization. Businesses invest in IT when they understand the competitive advantage and value added by the technology.

The aim of business modeling is to first establish a better understanding and communication channel between business engineering and software engineering. Understanding the business means that software engineers must understand the structure and the dynamics of the target organization (the client), the current problems in the organization and possible improvements. They must also ensure a common understanding of the target organization between customers, end users and developers.

Business modeling explains how to describe a vision of the organization in which the system will be deployed and how to then use this vision as a basis to outline the process, roles and responsibilities.

Requirements discipline

The goal of the Requirements is to describe what the system should do and allows the developers and the customer to agree on that description. To achieve this, analysts elicit, organize, and document required functionality.

Analysis and design discipline

The goal of analysis and design is to show how the system will be realized in the implementation phase. The aim is to build a system that:

- Performs—in a specific implementation environment—the tasks and functions specified in the use-case descriptions.
- Fulfills all its requirements.
- Is easy to change when functional requirements change.

Analysis and design results in a design model and optionally an analysis model. The design model serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written. The design model consists of design classes structured into packages and subsystems with well-defined interfaces, representing what will become components in the implementation. It also contains descriptions of how objects of these design classes collaborate to perform use cases.

Implementation discipline

The purposes of implementation are:

- To define the organization of the code, in terms of implementation subsystems organized in layers.
- To implement classes and objects in terms of components (source files, binaries, executables, and others).
- To test the developed components as units.
- To integrate the results produced by individual implementers (or teams), into an executable system.

Systems are realized through implementation of components. The process describes how you reuse existing components, or implement new components with well defined responsibility, making the system easier to maintain, and increasing the possibilities to reuse.

Test discipline

The purposes of the Test discipline are:

- To verify the interaction between objects.
- To verify the proper integration of all components of the software.
- To verify that all requirements have been correctly implemented.
- To identify and ensure that defects are addressed prior to the deployment of the software

The Rational Unified Process proposes an iterative approach, which means that you test throughout the project. This allows you to find defects as early as possible, which radically reduces the cost of fixing the defect. Tests are carried out along four quality dimensions *reliability, functionality, application performance, and system performance*. For each of these quality dimensions, the process describes how you go through the test lifecycle of planning, design, implementation, execution and evaluation.

Deployment discipline

The purpose of deployment is to successfully produce product releases, and deliver the software to its end users. It covers a wide range of activities including:

- Producing external releases of the software
- Packaging the software
- Distributing the software
- Installing the software
- Providing help and assistance to users

Although deployment activities are mostly centered around the transition phase, many of the activities need to be included in earlier phases to prepare for deployment at the end of the construction phase.The *Deployment and Environment* workflows of the Rational Unified Process contain less detail than other workflows.

Configuration and Change management discipline

The Change Management discipline in RUP deals with three specific areas:

- Configuration management
- Change request management
- Status and measurement management

Configuration management

Configuration management is responsible for the systematic structuring of the products. Artifacts such as documents and models need to be under version control and these changes must be visible. It also keeps track of dependencies between artifacts so all related articles are updated when changes are made.

Change request management

During the system development process many artifacts with several versions exist. CRM keeps track of the proposals for change.

Status and measurement management

Change requests have states such as new, logged, approved, assigned and complete. A change request also has attributes such as root cause, or nature (like defect and enhancement), priority etc. These states and attributes are stored in database so useful reports about the progress of the project can be produced. Rational also has a product to maintain change requests called ClearQuest. This activity has procedures to be followed.

Project management discipline

Project planning in the RUP occurs at two levels. There is a coarse-grained or **Phase** plan which describes the entire project, and a series of fine-grained or **Iteration** plans which describe the iterations.

However, this discipline of the Rational Unified Process (RUP) does not attempt to cover all aspects of project management. For example, it does not cover issues such as:

- Managing people: hiring, training, coaching
- Managing budget: defining, allocating, and so forth
- Managing contracts, with suppliers and customers

This discipline focuses mainly on the important aspects of an iterative development process:

- Risk management
- Planning an iterative project, through the lifecycle and for a particular iteration
- Monitoring progress of an iterative project, metrics

Project management discipline contain a number of other **Plans** and **Artifacts** that are used to control the project and monitoring its performance such **Plans** are:

- **The Phase Plan** (The Software Development Plan)
- **The Iteration Plan**

Phase plan

Each **Phase** is treated as a project, controlled and measured by the **Software Development Plan** which is grouped from a subset of monitoring plans:

- The **Measurement Plan**
  defines the measurement goals, the associated metrics, and the primitive metrics to be collected in the project to monitor its progress.
- The **Risk Management Plan**
  details how to manage the risks associated with a project. It details the risk management tasks that will be carried out, assigned responsibilities, and any additional resources required for the risk management activity. On a smaller scale project, this plan may be embedded within the Software Development Plan.
- The **Risk list**
  is a sorted list of known and open risks to the project, sorted in decreasing order of importance and associated with specific mitigation or contingency actions.
- The **Problem Resolution Plan** describes the process used to report, analyze, and resolve problems that occur during the project.
- The **Product Acceptance Plan**
  describes how the customer will evaluate the deliverable artifacts from a project to determine if they meet a predefined set of acceptance criteria. It details these acceptance criteria, and identifies the product acceptance tasks (including identification of the test cases that need to be developed) that will be carried out, and assigned responsibilities and required resources. On a smaller scale project, this plan may be embedded within the Software Development Plan.

Iteration plan

The iteration plan is fine-grained plan with a time-sequenced set of activities and tasks, with assigned resources, containing task dependencies, for the iteration.

There are typically two iteration plans active at any point in time.

- The **current iteration plan** is used to track progress in the current iteration.
- The **next iteration plan** is used to plan the upcoming iteration. This plan is prepared toward the end of the current iteration.

Therefore there are often two such plans: one for the **current iteration** and one under construction for the **next iteration**.

To define the contents of an iteration you need:

- the project plan
- the current status of the project (on track, late, large number of problems, requirements creep, and so on.)
- a list of scenarios or use cases that must be completed by the end of the iteration
- a list of risks that must be addressed by the end of the iteration
- a list of changes that must be incorporated in the product (bug fixes, changes in requirements)
- a list of major classes or packages that must be completely implemented

These lists must be ranked. The objectives of an iteration should be aggressive so that when difficulties arise, items can be dropped from the iterations based on their ranks.

Therefore there is a set of supported **Artifacts** that help in measuring and building each iteration plan

Work Product (Artifact)

IBM has replaced the term "artifact" with the term "work product". The work products used are:

- The **Iteration Assessment**
  captures the result of an iteration, the degree to which the evaluation criteria were met, lessons learned, and changes to be done.
- The **project measurements** is the project's active repository of metrics data. It contains the most current project, resources, process, and product measurements at the primitive and derived level.
- The **periodic Status Assessment**
  provides a mechanism for managing everyone's expectations throughout the project lifecycle to ensure that the expectations of all parties are synchronized and consistent.
- The **work order**
  is the Project Manager's means of communicating what is to be done, and when, to the responsible staff. It becomes an internal contract between the Project Manager and those assigned responsibility for completion.
- The **Issues List** is a way to record and track problems, exceptions, anomalies, or other incomplete tasks requiring attention.-

Environment discipline

The environment discipline focuses on the activities necessary to configure the process for a project. It describes the activities required to develop the guidelines in support of a project. The purpose of the environment activities is to provide the software development organization with the software development environment-both processes and tools-that will support the development team.

The Environment discipline workflow is broken down into three main steps:

Prepare Environment for Project
Preparing the development environment for a project means turning the underlying development process into an enactable project-specific development process. This involves :

- defining how the project is going to use the configured development process.
- developing a development case describing deviations of the underlying process.
- qualifying artifact selections with timing and formality requirements.
- preparing project-specific assets, like guidelines and templates, according to the development case.
- producing a list of candidate tools to use for development.

Prepare Environment for an Iteration
The purpose of this workflow detail is to ensure that the project environment is ready for the upcoming iteration. This includes process and tools.

This work is focused mainly on:

- Complete the Development Case to get ready for the iteration.
- Prepare and, if necessary, customize tools to use within the iteration.
- Verify that the tools have been correctly configured and installed.
- Prepare a set of project-specific templates and guidelines to support the development of *project artifacts in the iteration.
- Make sure that all the changes made to the project environment are properly communicated to *the project members

Support Environment During an Iteration
support the developers in their use of tools and process during an iteration.This includes installation of required software, ensuring that the

hardware is functioning properly and that potential network issues are resolved without delays.

The IBM Rational Method Composer product

The IBM Rational Method Composer (RMC) (http://www-306.ibm.com/software/awdtools/rmc/features/index.html) is a commercial product (built on top of Eclipse) for authoring, configuring, viewing, and publishing processes.

The RUP process framework within IBM Rational Method Composer includes:

- **A process content library**
  based on the best (or rather, least worst) practices adopted (endured) in thousands of RUP projects worldwide.
- **Out-of-the-box delivery processes**
  to provide the project manager with a quick starting point for planning and initiating a project. A delivery process will provide an initial project template, identify what milestones to have in the project, what work products to be delivered by each milestone, and what resources are needed for each phase.
- **Capability patterns**
  that allow project managers to rapidly add or remove reusable chunks of processes addressing common problems. Since no two projects are alike, project managers need to rapidly modify the process to address the specific project needs.

RMC has two main purposes:

- RMC is a content management system that provides a common management structure and look and feel for all process content. All content managed in RMC can be published to html and deployed to Web servers for distributed usage.
- RMC provides process engineers and project managers with the capability of selecting, tailoring, and rapidly assembling processes for their concrete development projects. RMC provides catalogs of pre-defined processes (like RUP) for typical project situations that can be adapted to individual needs. It also provides process building blocks called capability patterns that represent best development practices for specific disciplines, technologies, or development styles. These building blocks form a toolkit for quickly assembling processes based on project specific needs. Finally, the documented processes created with RMC can be published and deployed as Web sites. They can also be deployed as project plan templates for IBM Rational Portfolio Manager.

In additional to full RUP, RMC provides RUP plug-ins (extensions to the RUP content that can only be installed on top of RUP) for important areas such as service-oriented architectures (RUP for SOA), systems engineering (RUP SE), packaged application development (RUP for COTS (http://www-128.ibm.com/developerworks/rational/library/aug05/peraire-pannone/index.html)), program and portfolio management, etc. New delivery processes are added frequently and are made available via the IBM developerWorks Website (http://www-128.ibm.com/developerworks/rational/library/05/1206_ibmstaff/) .

There is an **open source version of IBM RMC** created as part of the Eclipse Process Framework (EPF) project. The EPF tool contains full process authoring and publishing capabilities. The main difference between EPF and the Rational Method Composer tool is the lack of integrations with other IBM Rational tools such as Rational Portfolio Manager and Rational Software Architect as well as lack of a migration capability from Rational Process Workbench. EPF comes with OpenUP/Basic, a new agile process for small teams applying RUP principles and practices.

Certification

In January 2007, the new RUP certification examination for *IBM Certified Solution Designer - Rational Unified Process 7.0* was released which replaces the previously called *IBM Rational Certified Specialist - Rational Unified Process*[2] (http://www-128.ibm.com/developerworks/rational/library/jan07/krebs/index.html) . -save- The new examination will not only test knowledge related to the RUP content but also to the process structure elements[3] (http://www-03.ibm.com/certify/certs/38008003.shtml)

Limitations

If the users of RUP do not understand that RUP is a process framework, they may perceive it as a weighty and expensive process. RUP was not intended, not envisioned and not promoted to be used straight “out of the box.” The IBM Rational Method Composer product has been created to address this limitation and help process engineers and project managers customize the RUP for their project needs. OpenUP/Basic, the lightweight and open source version of RUP, is another attempt to address this limitation.

As the RUP must be customized for each project by a RUP process expert, the project's overall success is highly dependent on the abilities of this one person.

See also

Refinements and variations

- Agile Unified Process
- Open Unified Process (OpenUP) - An open source software development process, created as part of the Eclipse Process Framework (EPF) project.
- OpenUP/Basic
  - The most agile and lightweight form of OpenUP, targets small and collocated teams interested in agile and iterative development.
- Enterprise Unified Process
- Essential Unified Process (EssUP)
- Unified Process - The generic Unified Process
- UPEDU - The Unified Process for Education

Alternative frameworks

Alternate approaches within the field of software engineering include:

- Cleanroom Software Engineering
- Dynamic Systems Development Method (DSDM)
- ICONIX Process is a lightweight, agile subset of the RUP practices
- Extreme Programming
- Scrum
- Harmony Process from Telelogic (was Ilogix) (http://www.ilogix.com/whitepaper-overview.aspx) covers both software engineering and system engineering
- Microsoft Solutions Framework (MSF)
- Tenstep Project Management

Software engineering

- Agile Software Development
- Software engineering
- Software development process
- Software component
- Project lifecycle
- Computer programming
- Quality assurance
- Extreme programming
- Agile Modeling
- Test-driven development
- Feature Driven Development

Books

- 2006: **Agility and Discipline Made Easy: Practices from OpenUP and RUP**

Per Kroll, Bruce MacIsaac [4] (http://www.awprofessional.com/title/0321321308)

- 2003: **Rational Unified Process Made Easy, The: A Practitioner's Guide to the RUP**

Per kroll [5] (http://www.awprofessional.com/title/0321166094)

- 1999: **The Unified Software Development Process**

Ivar Jacobson, Grady Booch, and James Rumbaugh [6] (http://www.amazon.com/gp/product/0201571692/)

- 1998: **The Rational Unified Process: An Introduction**

Philippe Kruchten [7] (http://www.amazon.com/gp/product/0201604590/) 2003: 3rd ed: [8] (http://www.amazon.com/gp/product/0321197704/)

External links

- Rational Software at IBM (http://www.rational.com/) .
- IBM Rational Unified Process Web Site (http://www-306.ibm.com/software/awdtools/rup/) .

- IBM Rational Method Composer Web Site (http://www-306.ibm.com/software/awdtools/rmc/features/index.html) .
- RUP Plug-Ins on IBM developerWorks Web Site (http://www-128.ibm.com/developerworks/rational/library/05/1206_ibmstaff/) .
- What Is the Rational Unified Process (http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf) - The Rational Edge, Jan 2001. (pdf)
- Key principles for business-driven development (http://www-128.ibm.com/developerworks/rational/library/oct05/kroll/) - The Rational Edge, Oct 2005.
- Implementing RUP/UP in 10 Easy Steps (http://www.x-tier.com/public/RUPUPIn10EasySteps.doc) - X-tier SAE website (doc).
- Understanding the Unified Process (http://www.methodsandtools.com/archive/archive.php?id=32) .
- Introducing IBM Rational Method Composer (http://www-128.ibm.com/developerworks/rational/library/nov05/kroll/index.html) - The Rational Edge, Nov 2005.
- IBM Rational Method Composer: Part 1: Key concepts (http://www-128.ibm.com/developerworks/rational/library/dec05/haumer/index.html) - The Rational Edge, Dec 2005.
- IBM Rational Method Composer: Part 2: Authoring method content and processes (http://www-128.ibm.com/developerworks/rational/library/jan06/haumer/index.html) - The Rational Edge, Jan 2006.
- The IBM Rational Unified Process for COTS-based projects: An introduction (http://www-128.ibm.com/developerworks/rational/library/aug05/peraire-pannone/index.html) - The Rational Edge, Aug 2005.
- The Eclipse Process Framework project (http://www-128.ibm.com/developerworks/rational/library/05/1011_kroll/index.html) - The Rational Edge, 2005.
- Eclipse Process Framework (EPF) Web site (http://www.eclipse.org/epf/) .

- Iterative Development and The Leaning Tower of Pisa (http://www.fromthetrench.com/2007/01/21/iterative-development-and-the-leaning-tower-of-pisa/) - From The Trench (http://www.fromthetrench.com/)

Categories: Software development process | IBM software | Project management

---