IBM Rational Method Composer: Part 1: Key concepts

Level: Introductory

Peter Haumer, Solution Architect, IBM

15 Dec 2005

from The Rational Edge: This two-part article offers a thorough introduction to IBM Rational Method Composer, IBM's next-generation process management tool platform and conceptual framework for authoring, tailoring, and deploying development processes. Part One, presented here, offers a high-level understanding of the key capabilities of RMC, and it provides an overview of key concepts and typical RMC usage scenarios.

For companies and organizations evolving into on demand businesses[1] to compete in a world increasingly enabled by multiple technologies and the disappearance of geopolitical boundaries -- "flatteners," as Thomas Friedman terms these enablers[2] -- the creation, integration, modernization, extension, and deployment of systems and software applications has become critical. Today, information technology not only supports the business but, in many organizations, actually defines the business.[3] We are witnessing the emergence of a new computing paradigm, which IBM calls "business-driven development"[4]; BDD organizations fully integrate and align flexible development processes with business process development for different lines of businesses as well as IT operations to improve business performance.

Running business-driven development projects requires very flexible development processes. Such processes not only have to provide concrete support and guidance for modern development practices, such as agile, iterative, architecture-centric, risk- and quality-driven software development, but also have to be flexible enough to support rapid tailoring and adoption of the process itself. These processes also need to evolve across projects, and the projects being executed must themselves be able to evolve as business needs change mid-way to completion.[5]

IBM Rational's Unified Process (RUP) has a long history as the leading commercial process framework product in the market, providing much of that flexibility through its architecture and tool support for the tailoring, extension, and deployment of processes.[6] This paper introduces the RUP development team's next-generation process management tool platform and conceptual framework -- IBM Rational Method Composer (RMC) -- which provides a completely redesigned user experience and a myriad of new features for authoring, tailoring, and deploying development processes.

This new platform consists of tools for authoring, configuring, and viewing method content and processes[7]. It replaces IBM Rational's current tools Rational Process Workbench, RUP Builder, and MyRUP. It also contains RUP-based method content and processes packaged in different variants. As of this writing, it is available as a commercial product offering and as the proposed framework called the "Eclipse Process Framework," as depicted in Figure 1:

1. *The Eclipse Process Framework (EPF)[8] is a proposed open-source project at eclipse.org. As proposed, IBM will donate major tool components and content from the next-generation RUP platform. As shown in Figure 1, 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 integration 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. The second part of this donation will include content supporting the new Basic Unified Process, a new agile process for small teams applying RUP principles and practices.*

2. *IBM Rational Method Composer (RMC) is IBM Rational's new commercial offering that includes everything donated for the EPF project, plus Rational tool integration and migration features as well as the full RUP content. In addition to the full version of RUP, RMC ships with RUP plug-ins (extensions to the RUP content that can only be installed on top of RUP) for important content areas such as service-oriented architectures, systems engineering, program and portfolio management, etc. Strictly speaking, this paper provides an overview of the Rational Method Composer tool platform being donated to the EPF project, but you can assume that all capabilities described apply to the both the EPF version as well as the full RMC tool unless differences are explicitly stated.*

This paper is presented in two parts. Part One offers a high-level understanding of the key capabilities of RMC, and it provides an overview of key concepts and typical RMC usage scenarios. Next month, Part Two will provide even more details about authoring method content and processes, including guidance and a review of the different forms of guidance supported by RMC.
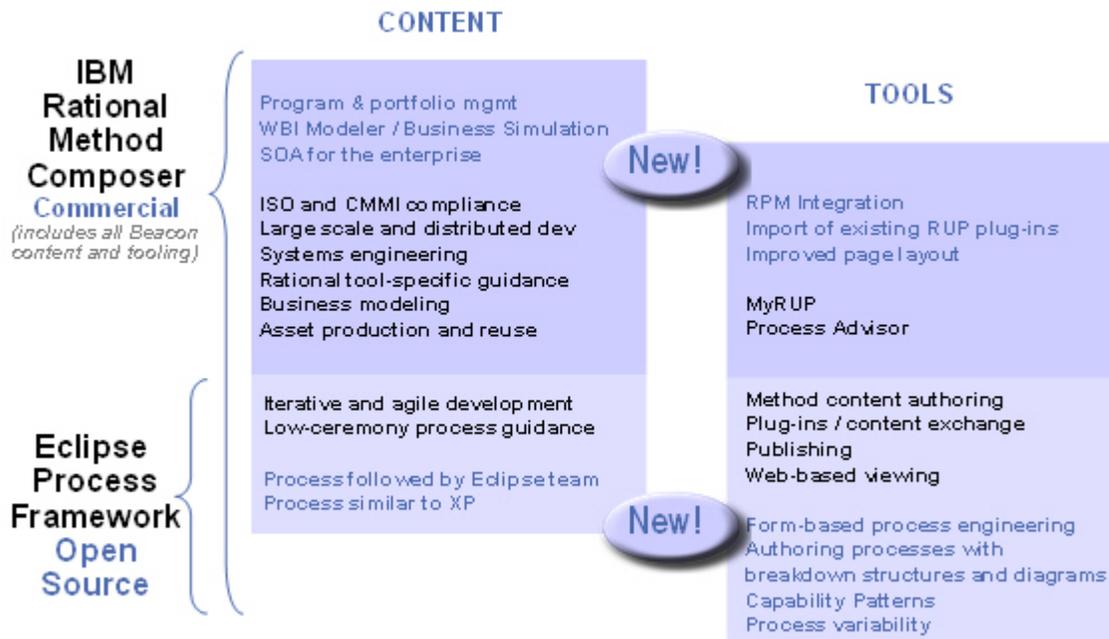

Figure 1: Method Composer versus Eclipse Process Framework overview. A black font in the two columns indicates features already present in past release. Blue font indicates features added for this release.

IBM Rational Method Composer: Its purpose and capabilities

Rational Method Composer (RMC) is a tool platform for process engineers, project leads, and project and program managers who are responsible for maintaining and implementing processes for development organizations or individual projects.

Typically, two key problems need to be addressed for such a process implementation. First, developers need to understand the methods of software development. They need to be familiar with the basic development tasks, such as how to elicit and manage requirements, how to do analysis and design, how to implement for a design or for a test case, how to test implementations against requirements, how to manage the project scope and change, and so on.

Although popular agile development methods rely on self-organizing teams[9] -- assuming that developers implicitly know how to do such work without documenting their methods -- many organizations still need to rely on explicitly documenting and educating such methods to establish common and regulated practices. In addition, many organizations are required to do so for compliance. (Note that RMC supports both of these audiences by making explicit method content optional as outlined in "Overview of key terminology and concepts.")

Second, development teams also need to define how to apply their development methods throughout a project lifecycle. That is, they need to define or select a development process. For example, requirements management methods have to be applied in one fashion during the early phases of a project, where the focus is more on elicitation of stakeholder needs and requirements and scoping a vision, and in a different fashion during later phases, where the

focus is on managing requirements updates and changes and performing impact analysis of these requirements changes. Teams also need a clear understanding of how the different tasks within the methods relate to each other: for example, how the change management method affects the requirements management method as well as the regression testing method throughout the lifecycle. Even self-organizing teams need to define a process that gives, at minimum, some guidance on how the development will be scoped throughout the lifecycle, when milestones will be achieved and verified, and so on.

RMC has two main purposes, which address the above needs:

1. To provide for development practitioners a knowledge base of intellectual capital that allows them to browse, manage, and deploy content. This content can be licensed, acquired, and, more importantly, accommodates your own content consisting of, for example, method definitions, whitepapers, guidelines, templates, principles, best practices, internal procedures and regulations, training material, and any other general descriptions of how to develop software. This knowledge base can be used for reference and education and forms the basis for developing processes (the second purpose). RMC is designed to be a content management system that provides a common management structure and look-and-feel for all of your content, rather than a document management system in which you would store and access hard-to-maintain legacy documents all in their own shapes and formats. All content managed in RMC can be published to HTML and deployed to Web servers for distributed usage.

2. To provide process engineering capabilities by supporting process engineers and project managers in selecting, tailoring, and rapidly assembling processes for their concrete development projects. RMC provides catalogs of predefined processes 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, an example of which is shown in Figure 2. RMC also allows you to set up your own organization-specific capability pattern libraries. Finally, the documented processes created with RMC can be published and deployed as Websites. In the commercial version of RMC, they can also be deployed as project plan templates for Rational Portfolio Manager.
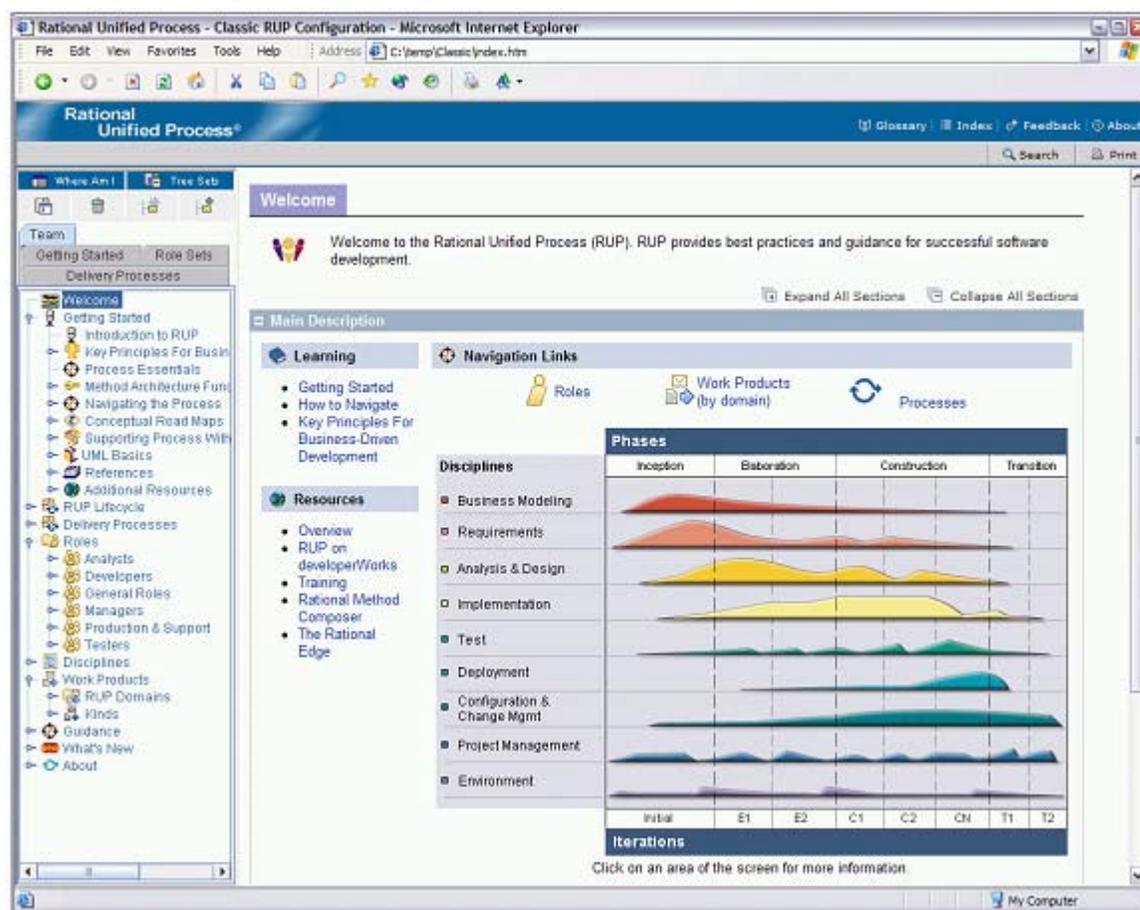
Figure 2: A Website published by Rational Method Composer containing and presenting all method and process content selected by the user for publication. The example shows a configuration of the IBM Rational Unified Process framework. Configurations of the Eclipse Process Framework or any user-generated content will appear in a similar way.

Rational Method Composer as well as the Eclipse Process Framework project aim at providing solutions to common problems that development leads and teams face when acquiring and managing their methods and processes. For example:

- *Development teams need easy and centralized access to the information*: Many development organizations do not maintain central databases of their practices and processes. Typically, processes are either not documented at all, or they are physically distributed in various presentation formats. Processes need to be deployed and accessible at the workplace, providing process documentation while the work is being performed.
- *It is difficult to integrate development processes that ship in their own propriety format*: Every book and publication presents method content and processes that use different formats. A lack of widely adopted standards and clearly defined concepts for representing method content and processes make it hard to integrate processes from different vendors and sources.
- *Teams lack an up-to-date knowledge base for educating themselves on methods and best practices*: Before effectively performing development processes, teams need to trained. They require a knowledge base or encyclopedia on development methods that consistently reflects the same practices on which processes are being defined and on which projects are being performed.
- *Teams need support for right-sizing their processes*: They need interactive guidance for answering the question of "How much process?" do they need. Processes need to be tailored not only for each project, but also continuously throughout the project lifecycle. Hence, processes need to be scaled up or scaled down on demand with the ability to easily assemble new or tailor existing processes to address organization-, project-, or even project-phase-specific needs.
- *Ensure compliance to standardized practices*: Teams need the ability to standardize on practices and processes within the organization, manage and deploy these process definitions, and provide the ability for individual projects to still perform auditable tailoring and modification of these processes.
- *Effective execution of processes in project*: Teams need to bridge the gap between process engineering and process enactment by using similar representation and terminology. Managers need the ability to import processes directly into their project execution environments, such as Rational Portfolio Manager, that link back from the planning elements, such as tasks to perform and their descriptions.

IBM Rational Method Composer replaces IBM Rational Process Workbench, RUP Organizer, and RUP Builder and provides the following new key capabilities:

- Provides completely redesigned tools for authoring, configuring, viewing, and publishing development processes.
- Provides just-in-time generation of publication previews in dedicated browsing perspective that allows rapid configuration switching.
- Manages method content using simple form-based user interfaces. Therefore, Unified Modeling Language (UML) skills are no longer required.
- Provides intuitive rich-text editors for creating illustrative content descriptions. Editors allow styles, images, tables, hyperlinks, and direct HTML editing.
- Allows creating processes with breakdown structure editors and workflow diagrams through use of multi-presentation process editors. Breakdown structure editor supports different process views: work-breakdown view, work-product-usage view, and team-allocation view. RMC automatically synchronizes all presentations with process changes.
- Is independent of the Rational Unified Process: Although the recent version of RUP is maintained and shipped with RMC, RMC is a general-purpose process engineering tool that provides support for many alternative lifecycle models. For example, waterfall, incremental, or iterative models can be created with the same overlapping method content.
- Improved reuse and extensibility capabilities. The plug-in mechanisms from past versions have been extended to support extensions for breakdown structures.
- Supports reusable dynamically-linked process patterns of best practices for rapid process assembly via drag-and-drop.

- Bridges the gap between process and project management by exporting processes into IBM Rational Portfolio Manager.

Overview of key terminology and concepts

To effectively work with Rational Method Composer, you need to understand a few concepts that are used to organize the content. This section provides a general overview of these concepts.

Method content versus process

The most fundamental principle in RMC is the separation of reusable core method content from its application in processes. This directly relates to the two purposes of RMC described in the previous section. Almost all of RMC's other concepts are categorized along this separation, as shown in Figure 6 (see farther below). Method content describes what is to be produced, the necessary skills required, and the step-by-step explanation describing how specific development goals are achieved. These method content descriptions are independent of a development lifecycle. Processes describe the development lifecycle. They take the method content elements and relate them into semi-ordered sequences that are customized to specific types of projects.
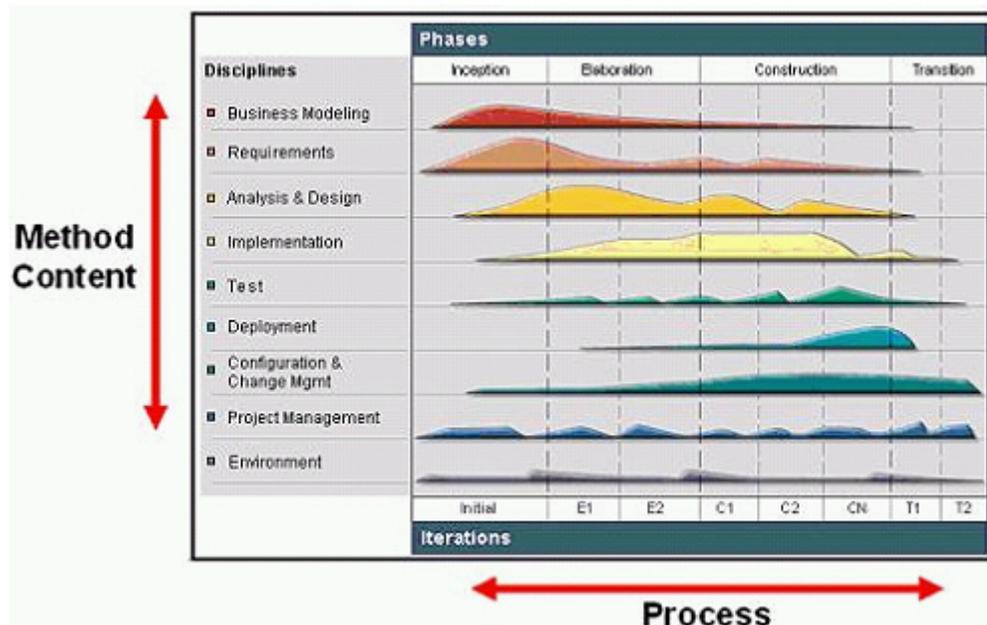
Figure 3: A two-dimensional representation of the Rational Unified Process's separation of method content and process.

This separation of method content and process is by no means a new idea. It has been present in the Rational Unified Process (RUP) since its inception and had already been described in 1992.[10] However with RMC, for the first time, a process tailoring tool shipping with RUP supports free modification of both these aspects. (Past versions of these tools focused on the method content side and offered limited authoring and tailoring capabilities for the process side.)

Figure 3 shows a two-dimensional representation of how this separation is depicted in RUP. Method content, describing how development work is being performed, is categorized by disciplines along the y axis. This work is then referenced and sequenced in a process along the x axis representing the timeline. This is the lifecycle of a development project; it expresses when specific work will be performed. The graphs in the middle represent an estimated workload for each discipline. As you can see, for example, one never stops working on requirements in RUP, but there are certainly peak times in which most of the requirements elicitation and description work is performed. There are also times at which a downward trend needs to be observed, where fewer requirements changes have to be processed to bring the project to a close. This avoids "feature creep," in which requirements work remains constant or even increases. Hence, a process expresses for such a lifecycle the variances of work being performed in the various disciplines, and the work itself is described by method content.

Method content

To learn about development methods, people do research in libraries or they receive training. Many development methods are described in books, articles, training material, standards and regulations, and other forms of documentation. These sources usually document methods in different ways, but mostly by providing step-by-step explanations for a particular way of achieving a specific development goal under general circumstances. Examples include transforming a requirements document into an analysis model; defining an architectural mechanism based on functional and nonfunctional requirements; creating a project plan for a development iteration; defining a quality assurance plan for functional requirements; redesigning a business organization based on a new strategic direction; and so on.

RMC takes such content and structures it in one specific way using a predefined schema. This schema is defined in the Unified Method Architecture (UMA),[11] which is an evolution of the RUP 2003 schema (also referred to as a meta-model), which was based on the SPEM 1.1 OMG standard.[12] UMA is also integrated with other IBM method schemata, such as IBM Global Services Method or IBM Rational Summit Ascendant. This schema supports organization of large amounts of descriptions for development methods and processes. Such method content and processes do not have to be limited to software engineering, but can also cover other design and engineering disciplines, such as mechanical engineering, business transformation, sales cycles, and so on.

Following this UMA schema, method content is represented in RMC as a construct of roles defining development skills and responsibilities for work products. These work products are produced by tasks that are performed by the roles and have work products as inputs and outputs. Figure 4 depicts typical sources for method content, as well as how the method content is represented in RMC. The RMC screen capture in Figure 4 shows how such method content elements are organized in tree browsers on the left. Similar to a library, these tree browsers provide different indexes of the available elements for rapid access. The screen capture shows on the right an example of a task presentation. This task presentation defines the task in terms of steps that need to be performed to achieve the task's purpose. You also see that the task has various relationships, such as relationships to roles that act as performers of the task and to work products that serve as inputs and outputs to the task. RMC maintains and presents these relationships as hyperlinks connecting elements to each other. (More details on tasks, roles, and work products will be provided in Part Two of this paper.) In addition to roles, tasks, and work products, RMC supports the addition of guidance elements. Guidance is supplementary free-form documentation such as whitepapers, concept descriptions, guidelines, templates, examples, and so on.

Figure 4: Method content can be found in books and publications on software engineering methods. Rational Method Composer expresses method content using concepts such as tasks, roles, work products, and guidance. This figure shows an example of how a task is presented in RMC.

Processes

A development process defines sequences of how work is being performed by roles and how work products are being produced and evolved over time. Figure 5 shows that processes are typically expressed as workflows or breakdown structures.
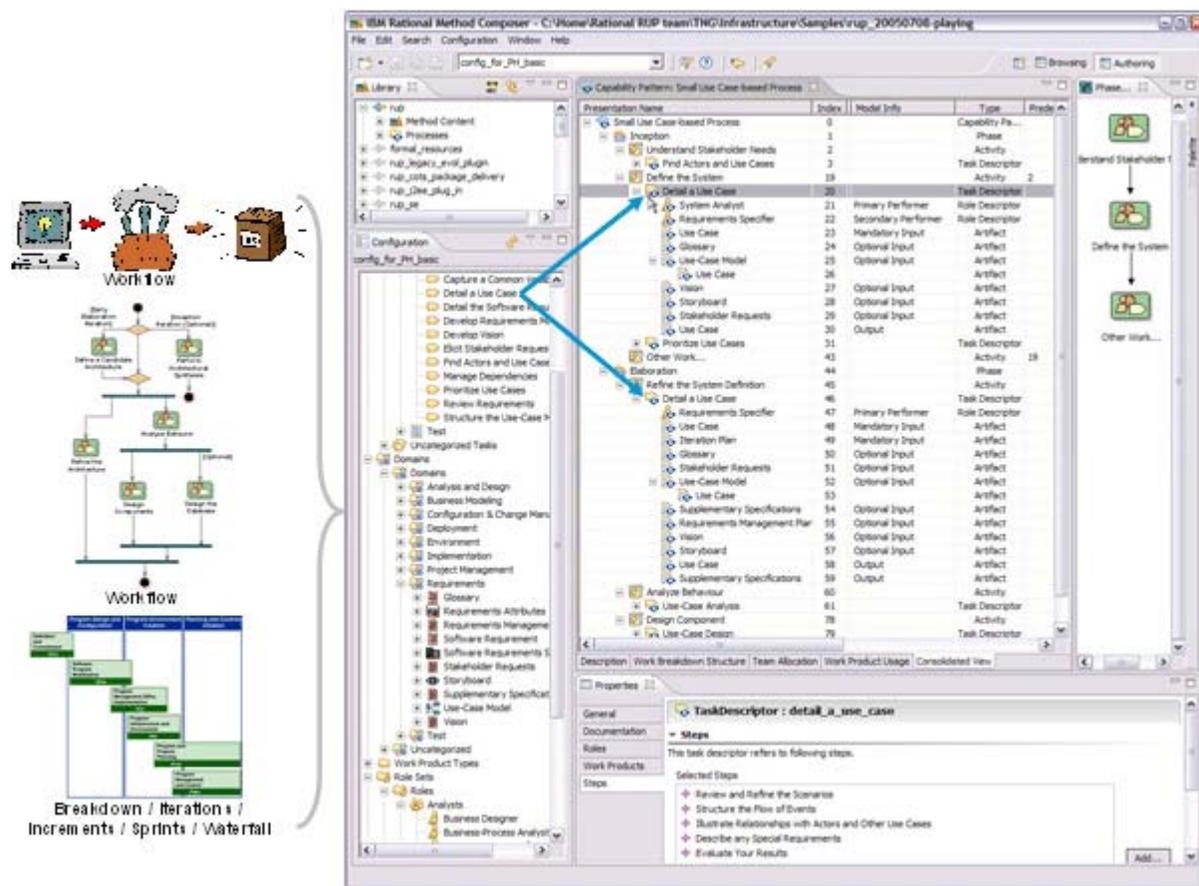
Figure 5: Processes are expressed as workflows or breakdown structures in RMC. They utilize method content and define how methods are being applied throughout the project lifecycle they describe.

Defining a strict sequence, as in a waterfall model, is just as much a process as defining a semi-ordered sequence of iterations in parallel work. These simply represent different development approaches.[13] In defining a process, one can take method content and combine it into structures that specify how the work shall be organized over time to support different development approaches, as well as to meet the needs of a particular type of development project, such as software for an online system versus software and hardware for an embedded system.

RMC supports processes based on different development approaches and can be used to define different lifecycle models such as waterfall, incremental, or iterative lifecycles. RMC also supports different presentations for processes, such as work breakdown structure or workflow presentations. You can also define processes in RMC that use a minimal set of method content or no method content at all to define processes for agile self-organizing teams (again, details will be provided in Part Two of this article).

The RMC screen capture on the right in Figure 5 shows an example of a process presented as a breakdown structure of nested activities. It also shows a workflow synchronized with this breakdown presented by an activity diagram for one particular activity: the Inception Phase. The figure also points out with the two blue arrows that the particular method content task, "Detail a Use Case," from Figure 4, has been applied in the process twice: first, in the Inception Phase under the activity "Define the System" and second, in the Elaboration Phase in the activity "Refine the System Definition." Below each of these task applications, referred to as task descriptors in RMC, you see lists of the performing roles as well as the input and output work products. If you look closely, you see that these lists are different for each of these two task descriptors, expressing differences in performing the detailing use-case methods throughout the lifecycle. You see different roles involved and changes in the list of inputs to be considered and outputs to be produced or updated. In this example, these changes were defined by the author who created this process to express the exact focus of the task performance for each occurrence at this particular point in the lifecycle. In addition to updating the roles plus input and output work products for a task descriptor, you can also provide additional textual descriptions and select the exact steps of the task that should or should not be performed in this particular occurrence of the task.

Summary

Figure 6 shows how the key RMC concepts relate to method content versus process separation. As you see, method content is primarily expressed using work products, roles, tasks, and guidance.
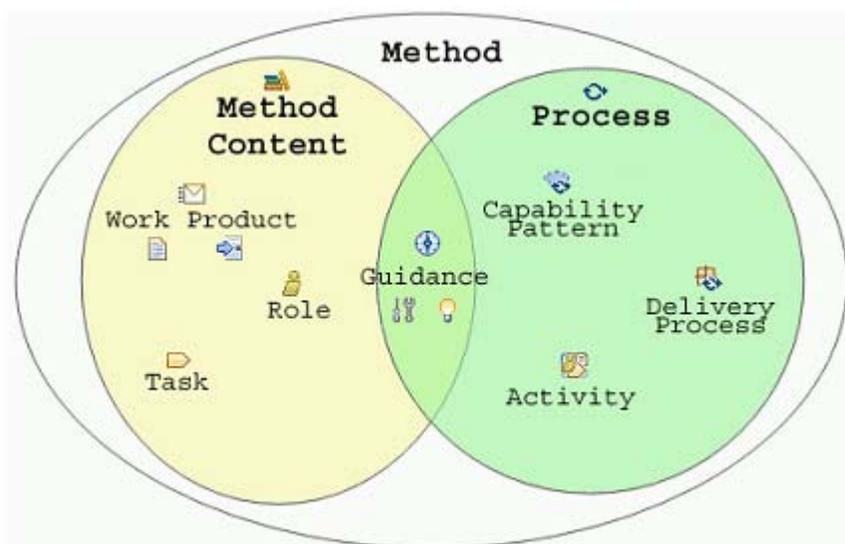

Figure 6: Terminology overview for RMC concepts.

Guidance, such as checklists, examples, or roadmaps, can also be defined to provide background and exemplary walk-throughs of a process. On the right-hand side of Figure 6, you see the concepts used to represent processes in RMC. The main concept is the activity that can be nested to define breakdown structures. Activities relate to each other to define the flow of work. They contain descriptors that reference method content, and they are used to define processes of which RMC supports two main kinds: delivery processes and capability patterns. Delivery processes represent a complete and integrated process template for performing one specific type of project. They describe a complete end-to-end project lifecycle and are used as a reference for running projects with similar characteristics. Capability patterns are processes that express process knowledge for a key area of interest, such as a discipline or a best practice. They can be directly used by process practitioners to guide their work. They are also used as building blocks to assemble delivery processes or larger capability patterns, thus ensuring optimal reuse and application of the key practices they express.

Typical usage scenarios for working with RMC

Let's have a look at the most typical usage scenarios for Rational Method Composer or the Eclipse Process Framework. Although RMC is a powerful process-authoring environment, there are various degrees of authoring and modes of utilizing RMC's capabilities. RMC contains a lot of method and processes content out of the box. Therefore, in many cases, not much authoring is actually required. All that is needed is selecting, configuring, and tailoring the existing content to fit your needs.

The simplest usage scenario for RMC involves simply using the processes as they ship -- out of the box. The commercial version of RMC contains the complete sources of the Rational Unified Process framework (RUP v7.0), which consists of thousands of method and process elements for a broad variety of development situations. It also includes various method plug-ins extending RUP with domain-specific additions such as development for concrete technologies such as Java Enterprise Edition (JEE) or different development circumstances such as adopting a commercial-off-the-shelf system (COTS). The Eclipse Process Framework contains a rapidly growing library of content donated by the project's committers.

However, be aware that no organization or project requires all of this documentation all at once. Instead, you should work with a specific subset called a method configuration, which tailors the process for specific needs. During the selection and tailoring activities, you still might want to include your own content and link that into the existing available material. This is where RMC's authoring capabilities provide you with easy-to-use but powerful editors to do this in a seamless way.

You could also start from scratch in RMC and not rely on any explicit method definitions that we provide. All the process authoring done in this case would consist of defining your lifecycle structure using releases, phases, iterations, sprints, or however you want to organize your lifecycle, and populating this lifecycle structure with high-level activities, milestones, and perhaps definitions for key deliverables.

The following subsections summarize the most common usage scenarios for RMC.[14]

Selecting and configuring existing method content and processes

This represents one of the simplest RMC usage scenarios. RUP 2003 users will recognize it as the "RUP Builder" scenario. You select the process and underlying method content that fits your needs by browsing the RMC method library, which contains all of your purchased content as well as content that you downloaded from the RMC resource center[15] or the Eclipse Process Framework community. Once you find a close fit, you start configuring this process (as depicted in Figure 7) by selecting or deselecting what we call "method packages." Removing a method package removes all references to the content of this package from everywhere in the process. You can, for example, "strip down" a process to contain a minimal subset of its content by removing packages that contain elements of work that you do not want to perform. Or, you could add method packages with very specific content for a particular domain that this process supports as an option (for example, "plain" JEE versus JEE for SOA-specific content).
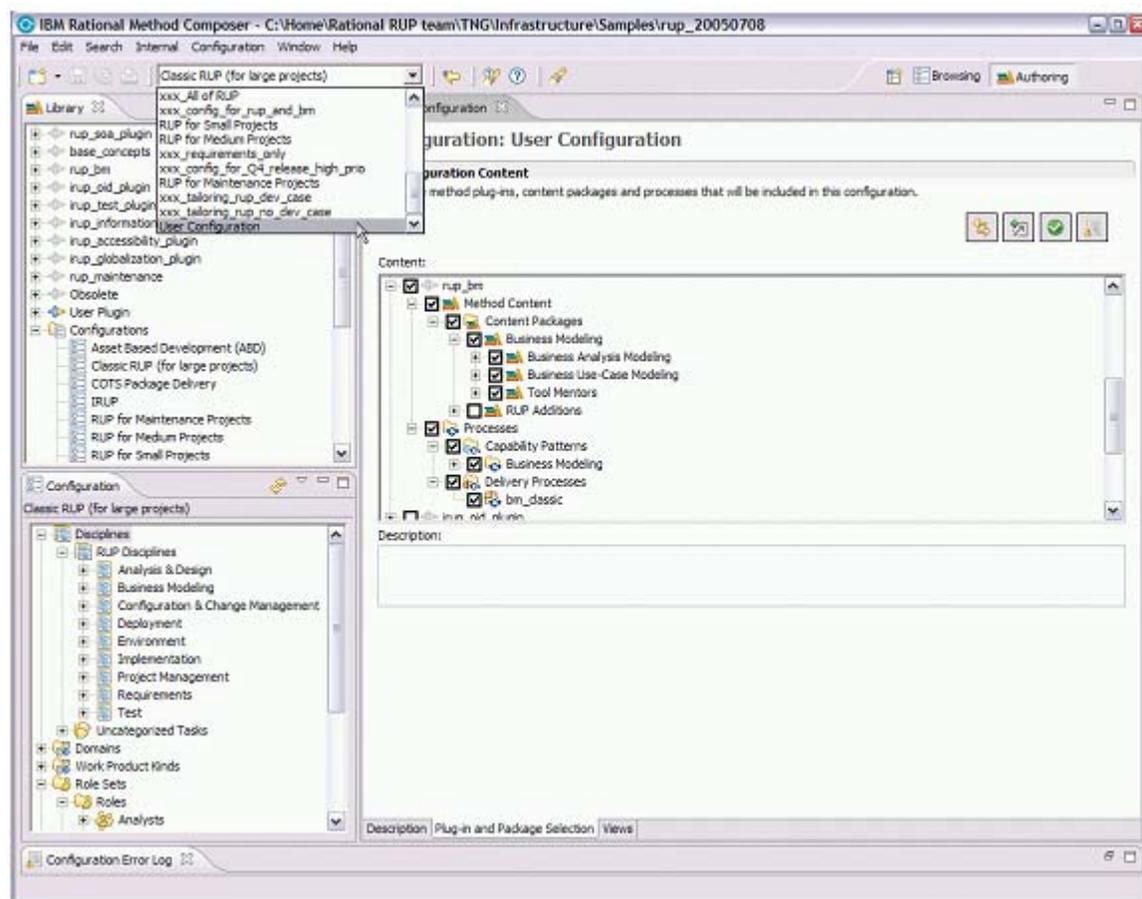

Figure 7: Configure processes and method content by selecting or deselecting method packages that shall be referenced or not referenced by the processes. You can create several method configurations defining filters on the overall method library in RMC. Use the combo box in the toolbar to quickly switch between different method configurations. The Configuration view in the lower-left corner always displays the resulting contents of a method configuration.

Method content and processes are organized in RMC according to the way they build logical units for useful configurations. For example, all content belonging to one specific discipline, such as requirements or change management, can be found in one method package. Each of these packages might be further divided into subpackages for specific practices in these disciplines. For example, the requirements discipline in RUP factors all of its use-case

related content into a separate package. Thus, you can add or remove just use-case practices or the whole requirements management discipline from your process with just one simple mouse-click by selecting or deselecting the use-case method or requirements package.

The result is a configured process that you then publish and deploy to your team in HTML (see Figure 2 for an example) and/or export into your project management tool. Note that you could create such method configurations for processes that span the whole development lifecycle as well as just one or more phases, iterations, activities, and so on. Hence, you are not forced to define your entire lifecycle up front. Instead, you can configure your process iteratively, as needed.

Tailor an existing process

In addition to simply configuring a process, you can actively modify processes to make them conform even better to your specific needs by using one or more of RMC's editors. You can create what we call a process contribution for an existing process that defines differential changes to that process. You can directly add, remove, or replace elements in the process. Hence, in contrast to changing the configuration -- which would make global changes on the process -- you can define individual changes just in the places you require them. RMC provides sophisticated dynamic linking capabilities that separate your changes from the original process definition, so that when the underlying process changes, you can simply upgrade without losing your changes. (In Part Two, I will provide examples on process tailoring.)

Create a new process

As an alternative to tailoring an existing process, you can also author a completely new process from scratch or a new process that reuses material from one or more existing processes. In cases where you cannot find any reusable material at all, you can create a completely new process from scratch. In most cases, however, you will start developing your own process by assembling reusable building blocks from method content as well as predefined process patterns that we call capability patterns. A capability pattern is a mini process defining reusable clusters of activities, which are typically performed over and over again, in a common process area. Examples of capability patterns include "use-case-based requirements management," "develop components," "validate build," or "ongoing management and support." In addition to replicating such patterns each time the work described by the pattern will be performed, RMC allows you to dynamically link patterns into your process. When the pattern changes, all applications of the pattern will be automatically updated.

RMC's process-authoring capabilities let you fully utilize the separation of method content and process as outlined in "Overview of key terminology and concepts." You can start creating your process by defining your own lifecycle model and then systematically populating it with method content and/or capability patterns that you either define yourself or reuse from the method library.

Figure 8 depicts an example from the Eclipse Process Framework, demonstrating how method content and capability patterns have been reused in two processes with different lifecycle models.

Figure 8: Two processes with two different lifecycle models applying the same capability patterns and method content.

In Figure 8, the right-hand side contains two breakdown structure process editors. The top one defines the Basic Unified Process (BUP), which is an agile lightweight adaptation of RUP.[16] The bottom one defines a Scrum-like process. Each of the two has its own distinct lifecycle model. BUP uses the four RUP phases -- Inception, Elaboration, Construction, and Transition -- and defines iterations for them. Scrum processes are organized in iterations also referred to as sprints. If you carefully examine the two processes in Figure 8, you see that there are commonalities as well as process-specific differences. Activities such as Manage Iteration or Manage Sprint list different tasks because both processes have a different management approach. They have been authored specifically for each process.

On the other hand, both processes share some activities, such as Develop Solution or Validate Build.

These activities represent capability patterns that have been defined as basic building blocks for the Basic Unified Process; as you can see, they are listed in the tree browser on the left of Figure 8. These patterns have been reused by the Scrum process author who applied them to the process by simple drag-and-drop operations. Devoted Scrum practitioners might say that this process violates the principles of Scrum by providing explicit tasks for doing work instead of assuming self-organizing teams. However, we created this example to show you how processes like Scrum can easily be enriched with RMC's process-authoring capabilities by just reusing patterns and tasks that could serve as guidance rather than strict work instructions for perhaps less experienced teams.

Develop method content and create or extend processes

The last scenario presented here is the ability to not only create or tailor processes reusing RMC or third-party method content, but also to develop your own method content and use that either to tailor existing processes or use in the

creation of new processes.

For method content authoring, you again have the choice of defining completely new content or extending existing method content. You develop new method content if you need to define your own roles, want to add additional work product types, want to add your development methods, or simply want to provide additional guidance -- such as whitepapers, your organization's specific regulations and guidelines, or your own work product templates or checklists. If you simply want to modify existing method content available in the method library by adding a work product responsibility to a role, adding steps to a task, or adding a few more check points to an existing checklist, you can do so with RMC's unique method plug-in and variability capabilities. Variability allows you to change existing content without directly modifying the original. You have, for example, the ability to create a method plug-in to RUP that contains a task contribution adding some new steps to an existing task. Or, you could define in your plug-in a replacing work product for a RUP work product that defines your own variant. For example, it might have a different name, structure, and templates as well as customized descriptions and guidance. All references other RUP elements had to the original work product will be automatically replaced with references to your element. You can then optionally switch on and off your extensions using method configurations as well as easily upgrade to newer versions of the original elements and reapply your changes.

In RMC, you define and document method content with intuitive form-based editors that are easy to learn, but powerful enough to provide well-formatted rich text documentation.

Figure 9 shows the editor for a work product definition. To define a new work product type, you just create such an element in a method package. To document it, you just need to use its editor for filling in form fields and creating relationships using selection dialogs and combo boxes. RMC creates and manages the HTML behind the scenes that provides the well-formatted and hyperlinked documentation that you saw in Figure 4 (showing a preview of page documenting a task).
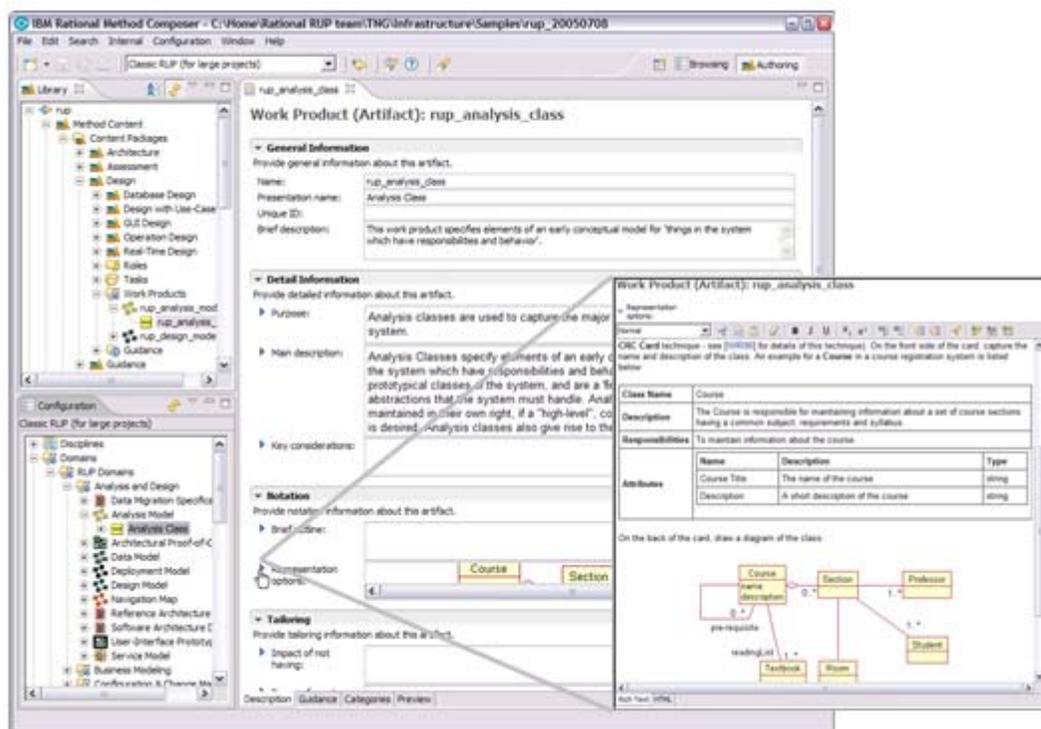

Figure 9: Form-based editor for a work product. Every form field can be expanded into a full-sized rich text editor that allows you to format text, work with tables, include images, and so on.

Once you have created your own method content elements, you can include them in your method configurations and add them to your own or reused processes using the "Tailor Existing..." or "Create New..." scenarios described above.

This concludes our overview of the most common RMC usage scenarios. Next month, I will present a more in-depth look at method content and process in Part Two of this paper. See you then!

Acknowledgements

Notes

[1] See IBM, "On Demand Business," December 2005: http://www.ibm.com/ondemand. Also, Alfredo Gutierrez, "e-business on demand: A developer's roadmap," IBM developerWorks, February 2003: http://www-128.ibm.com/developerworks/ibm/library/i-ebodov/index.html

[2] Thomas L. Friedman, *The World is Flat: A Brief History of the 21st Century*. Farrar, Straus and Giroux 2005.

[3] Asiff Hirji, CIO Ameritrade, Gartner Financial Services Technology Summit, Aug. 2005: http://www.computerworld.com/careertopics/careers/story/0,10801,104482,00.html

[4] Per Kroll and Walker Royce, "Key principles for business-driven development," *The Rational Edge*, October 2005: http://www-128.ibm.com/developerworks/rational/library/oct05/kroll/index.html

[5] Per Kroll and Walker Royce, "Key principles for business-driven development," *The Rational Edge*, October 2005: http://www-128.ibm.com/developerworks/rational/library/oct05/kroll/index.html

[6] For more on RUP, see the following four sources: IBM, "Rational Unified Process," version 2003.06, IBM Rational Software, 2003; IBM, "Rational Unified Process," version 7.0, IBM Rational Software, 2006; Philippe Kruchten, *The Rational Unified Process: An Introduction*. Addison Wesley 2000; and Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process*. Addison Wesley 2003.

[7] For a general overview of what is included in RMC, see Per Kroll, "Introducing IBM Rational Method Composer," *The Rational Edge*, November 2005: http://www-128.ibm.com/developerworks/rational/library/nov05/kroll/index.html

[8] See Eclipse.org, "The Eclipse Process Framework. Project Beacon," Eclipse.org Proposal, 2004: http://www.eclipse.org/proposals/beacon/. See also Per Kroll, "The Eclipse Process Framework project," IBM developerWorks, November 2005: http://www-128.ibm.com/developerworks/rational/library/05/1011_kroll/; see also Ricardo Balduino, "Basic Unified Process: A Process for Small and Agile Projects," Eclipse.org 2005: http://www.eclipse.org/proposals/beacon/Basic%20Unified%20Process.pdf

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

[10] I. Jacobson et al., *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley 1992.

[11] For a detailed introduction to UMA, which has been submitted to the OMG for the next version of the SPEM standard, see Peter Haumer and Todd Fredrickson, "An Overview to the Unified Method Architecture," IBM, to be published. For more information on the next SPEM standard, see OMG, "Software Process Engineering Metamodel," version 2.0, RFP, 2004: http://www.omg.org/cgi-bin/doc?ad/2004-11-4

[12] OMG, "Software Process Engineering Metamodel," version 1.1, formal/2005-01-06, 2005: http://www.omg.org/technology/documents/formal/spem.htm

[13] For an in-depth comparison of these two approaches, see Walker Royce, *Software Project Management: A Unified Framework*. Addison Wesley Longman 1998.

[14] For a more detailed discussion on tailoring RUP, see the "Tailor RUP" section of the IBM Rational Unified Process, version 7.0, IBM Rational Software, 2006.

[15] See IBM, "RMC resource center on IBM developerWorks," 2006:
http://www-128.ibm.com/developerworks/rational/products/rup/

[16] See Ricardo Balduino, Op cit.

About the author

Peter Haumer is a solution architect for the IBM Rational Method Composer product platform. He is responsible for defining next-generation process engineering tools, and he represents IBM at the OMG in the SPEM 2.0 initiative. Before joining the Rational Unified Process team, he worked as a senior professional services consultant for the IBM Rational brand. He performed on-site consulting and training, and he assisted and coached customers to be successful with the Rational Unified Process and other Rational tools. His areas of interest include requirements management, object-oriented analysis and design for enterprise application architectures, and software process implementation. Before joining Rational, he worked in basic research in the areas of requirements engineering and flexible process-integrated CASE tool architectures. Peter received his doctorate in computer science from the Technical University Aachen, Germany.