Eclipse Process Framework Composer

Part 1: Key Concepts

Second Revision, April 2007

by Peter Haumer, phaumer@us.ibm.com
Solution architect, IBM Rational Software

This two-part article offers an introduction to the process engineering approach and tool support provided by the Eclipse Process Framework (EPF). It presents a 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 the Eclipse Process Framework Composer tool and it provides an overview of key concepts and typical usage scenarios.

Keywords: Eclipse Process Framework, process engineering, project management, software development process, requirements management, configuration management, agile development, software development best practices, IBM Rational Method Composer, Rational Unified Process, SPEM.

Author bio: Dr. Peter Haumer is an Eclipse Process Framework committer. He is a solution architect for the IBM Rational Method Composer product platform and responsible for defining next generation process engineering tools. He represents IBM at the OMG in the SPEM 2.0 initiative.

For companies and organization 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, often referred to as "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 a 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]

This paper introduces the Eclipse Process Framework Composer (EPF Composer), which addresses the needs for BDD as well as modern agile development practices. The EPF Composer is a process management tool platform and conceptual framework, which provides an easy to learn user-experience and simple to use features for authoring, tailoring, and deploying of development process frameworks. The paper presents the EPF Composer version 1.0.2 that is available as a free download under the Eclipse

---

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

[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, http://www.computerworld.com/careertopics/careers/story/0,10801,104482,00.html, Aug. 2005.

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

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

Public license at the EPF homepage[6]. IBM also offers a commercial version of the EPF Composer tool named IBM Rational Method Composer[7] (RMC) that ships with the IBM Rational Unified Process (RUP) and provides additional enterprise capabilities not described in this article.

This paper is presented in two parts. Part One offers a high-level understanding of the key capabilities of the EPF Composer environment, and it provides an overview of key concepts and typical usage scenarios. Part Two will provide more specific details and examples about authoring method content and processes with EPF Composer.

[6] http://www.eclipse.org/epf/
[7] http://www-128.ibm.com/developerworks/rational/products/rup

1 Eclipse Process Framework Composer: What is it for?

Eclipse Process Framework Composer (EPF Composer) is a tool platform for process engineers, project leads, 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 and key practices 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[8] – assuming that developers implicitly know how to do such work without documenting their methods – and learning through tutoring, 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 EPF Composer supports both of these audiences by making explicit method content optional as outlined in Section 2.)

Second, development teams also need to define how to apply their development methods and best practices 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 impacts 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.

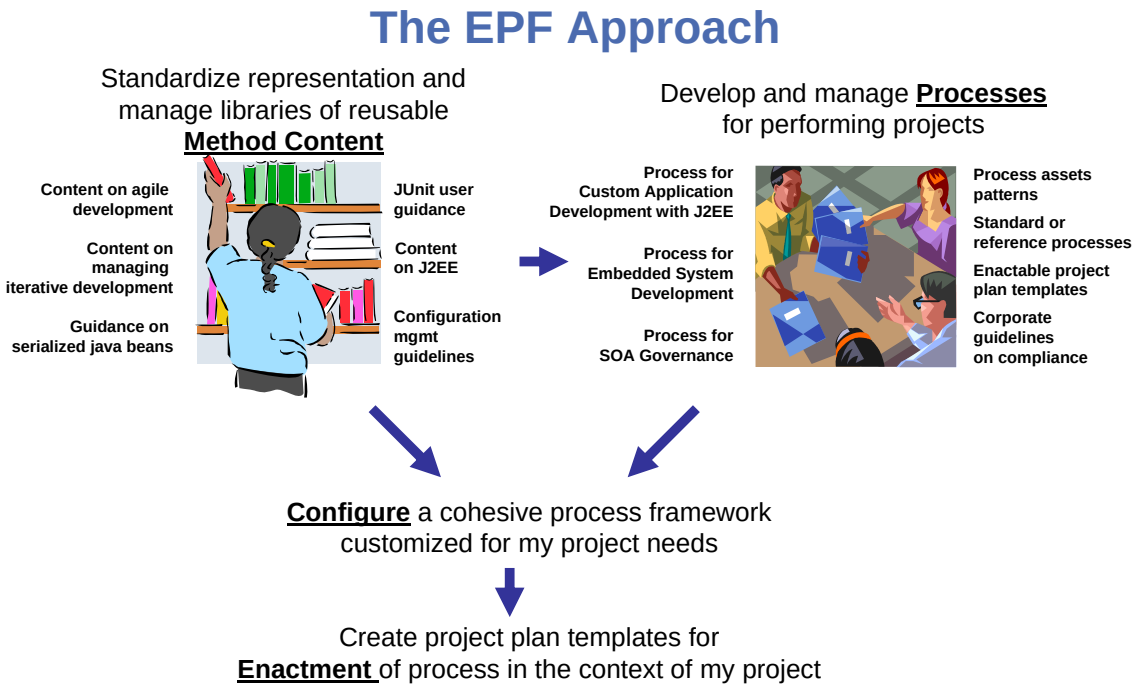

Figure 1: The EPF Approach of managing libraries of reusable method content that can be used to assemble processes. Reusable method content and processes can be configured by EPF Composer users for specific project needs and then published for enactment as project plans or process documentation.

EPF Composer 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). EPF Composer is designed to be a content management system that provides a common management structure

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

and look and feel for all of your content, rather than being 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 EPF Composer 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. EPF Composer provides catalogs of pre-defined 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. EPF Composer also allows you to set-up your own organization-specific capability pattern libraries. Finally, the documented processes created with EPF Composer can be published and deployed as Web sites.

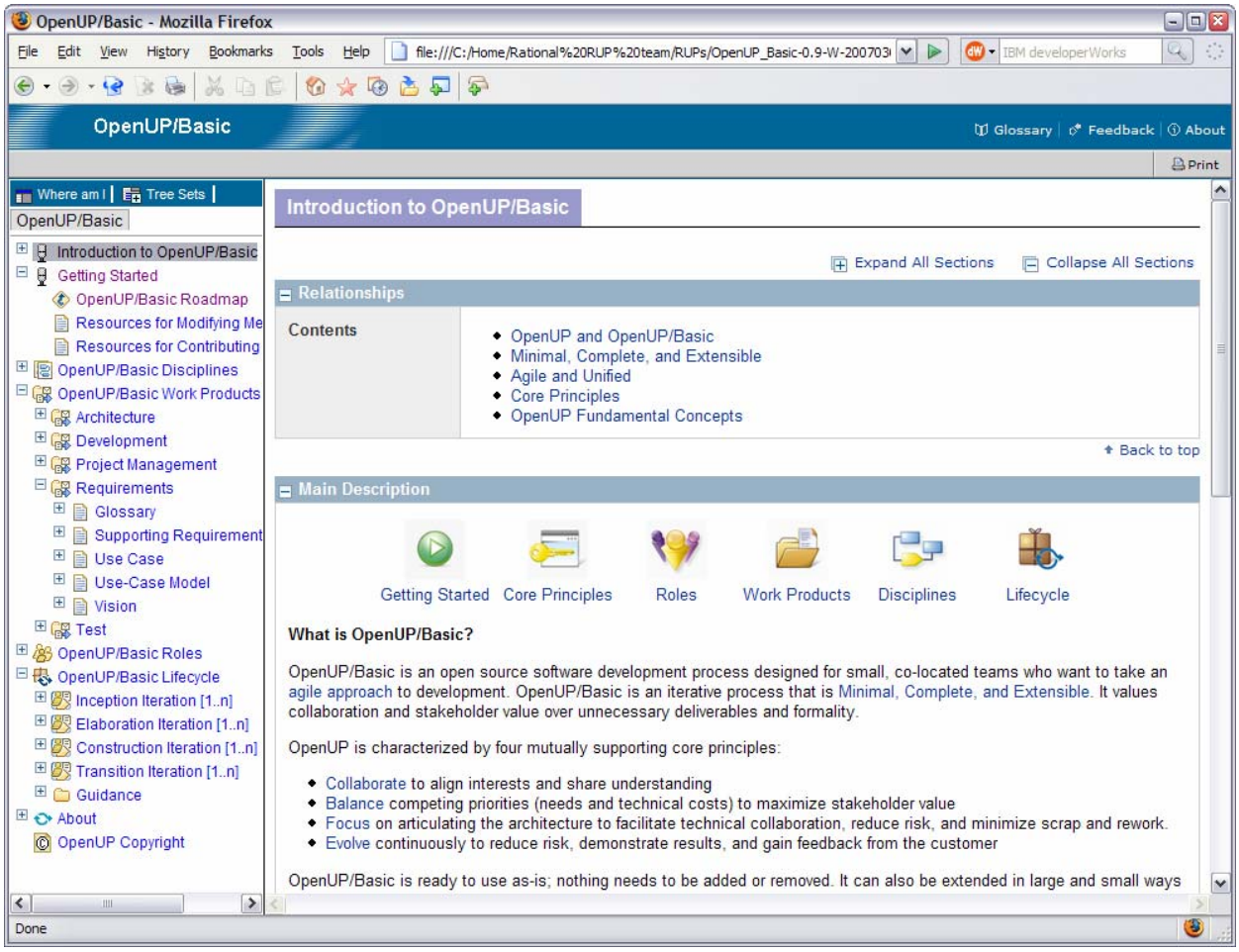

Figure 2: A Web site published by Eclipse Process Framework Composer containing and presenting all method and process content selected by the user for publication. The example shows the EPF OpenUP/Basic[9] framework.

The Eclipse Process Framework project aims 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 process use a different format. 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 be trained. They require a knowledge base or encyclopedia on development methods which consistently reflects the same practices on which processes are being defined 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 that link back from the planning elements, such as tasks to perform and their descriptions.

[9] See Ricardo Balduino, Brian Lyons, " OpenUP/Basic - A Process for Small and Agile Projects", at http://www.eclipse.org/epf/general/OpenUP_Basic.pdf, eclipse.org/epf, 2006.

2 Overview of key terminology and concepts

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

2.1 Method content versus process

The most fundamental principle in the Eclipse Process Framework is the separation of reusable core method content from its application in processes. This directly relates to the two purposes of EPF Composer described in the Section 1. Almost all of EPF Composer's other concepts are categorized along this separation, as shown in Figure 6 (see further 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.

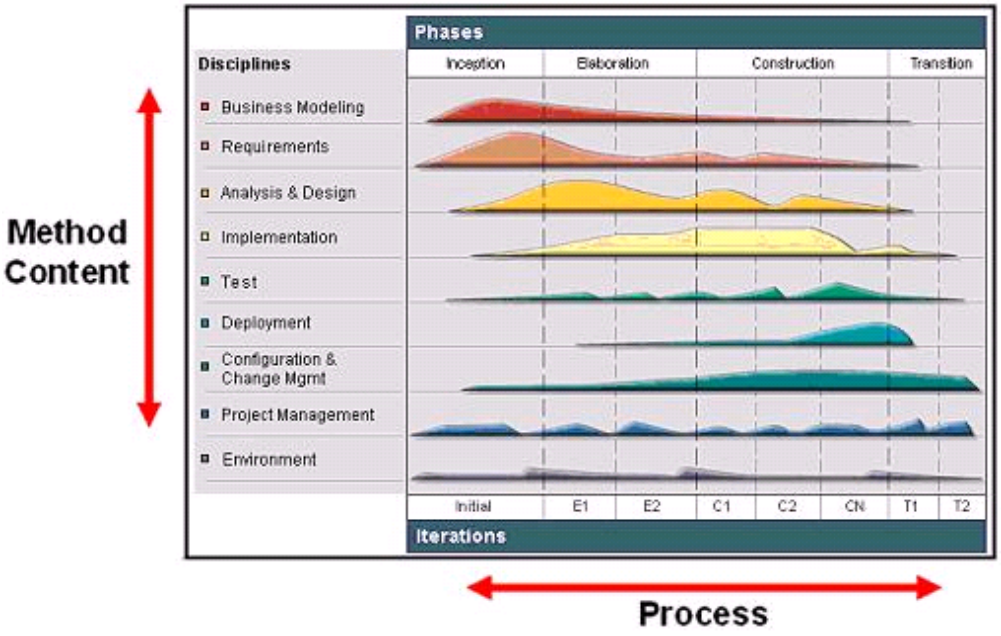

Figure 3: A two-dimensional representation of the Unified Process' separation of method content and process.

This separation of Method Content and Process is by no means a new idea. For example, it has been present in the Unified Process (UP) since its inception, also sometimes referred to as the static versus the dynamic structure[10], and had already been described for OOSE[11] in 1992.

Figure 3 shows a two-dimensional representation of how this separation is depicted in the Unified Process (UP)[12] as well as the Rational Unified Process (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 UP, 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 explicitly defined and described by method content.

2.2 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 non-functional requirements; creating a

[10] Philippe Kruchten, "The Rational Unified Process: An Introduction," Addison Wesley, 2003.
[11] I. Jacobson et al., "Object-Oriented Software Engineering: A Use Case Driven Approach," Addison-Wesley, 1992.
[12] Ivar Jacobson et al, "The Unified Software Development Process", Addison Wesley, 1998.

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.

EPF Composer allows you to take such content and to structure it in one specific way using a predefined schema. This schema is an evolution of the SPEM 1.1 OMG specification[13] referred to as the Unified Method Architecture (UMA). Major parts of UMA went into the recently adopted revision of SPEM, SPEM 2.0[14]. EPF is aiming to fully support the final SPEM 2.0 in the near future. The UMA and SPEM schemata support the 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 the UMA and soon SPEM 2.0 schema, method content is represented in EPF Composer 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 EPF Composer. The 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. EPF Composer 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,

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

[14] OMG, "Software Process Engineering Metamodel," version 2.0, Final Adopted Specification, ptc/07-03-03, http://www.omg.org/cgi-bin/doc?ptc/07-03-03.

tasks, and work products, EPF supports the addition of guidance elements. Guidance is supplementary free-form documentation such as whitepapers, concept descriptions, guidelines, templates, examples, and so on.

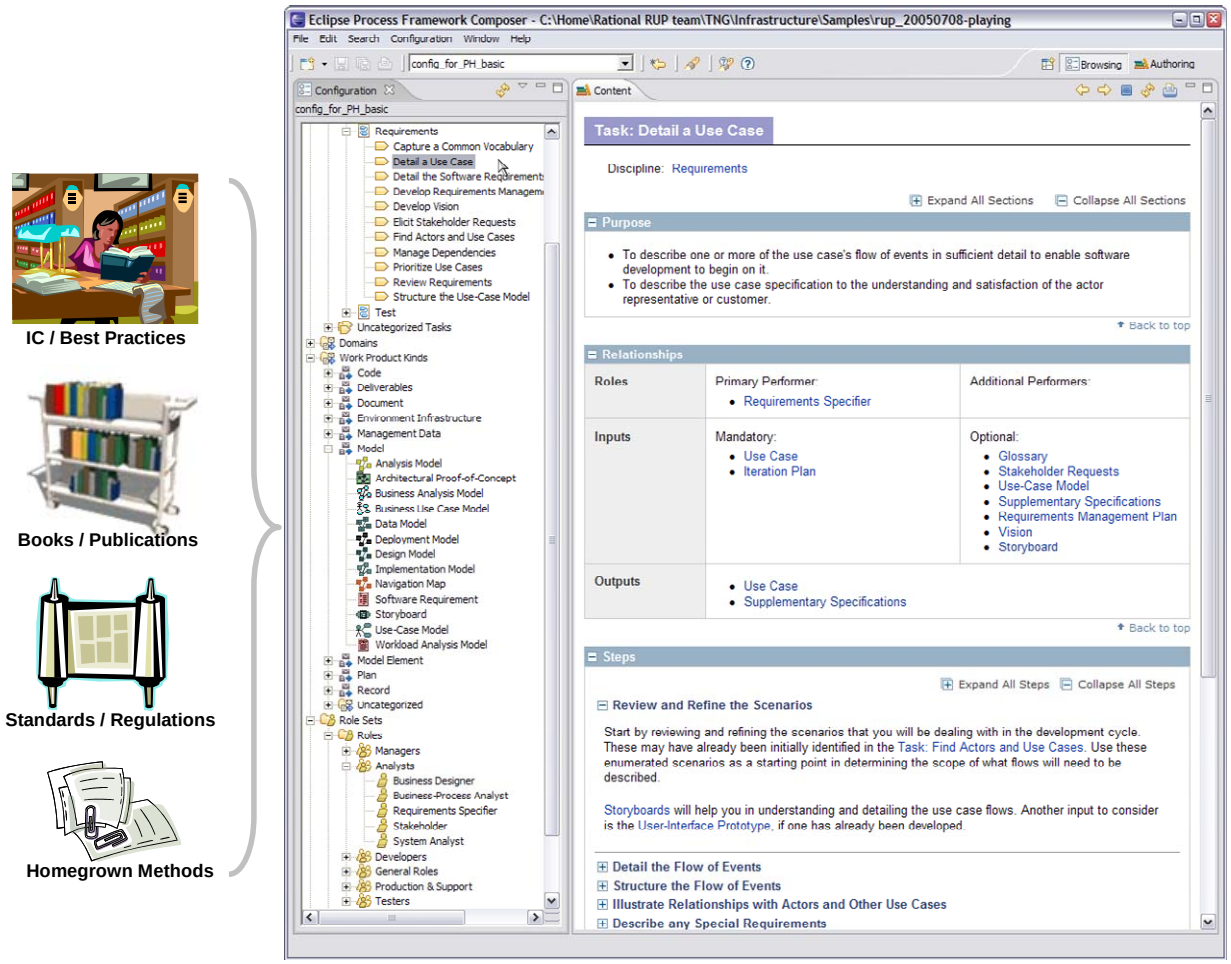

Figure 4: Method content can be found in books and publications on software engineering methods. Eclipse Process Framework Composer expresses method content using concepts such as tasks, roles, work products, and guidance. This figure shows an example of how a task is presented in EPF.

2.3 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.

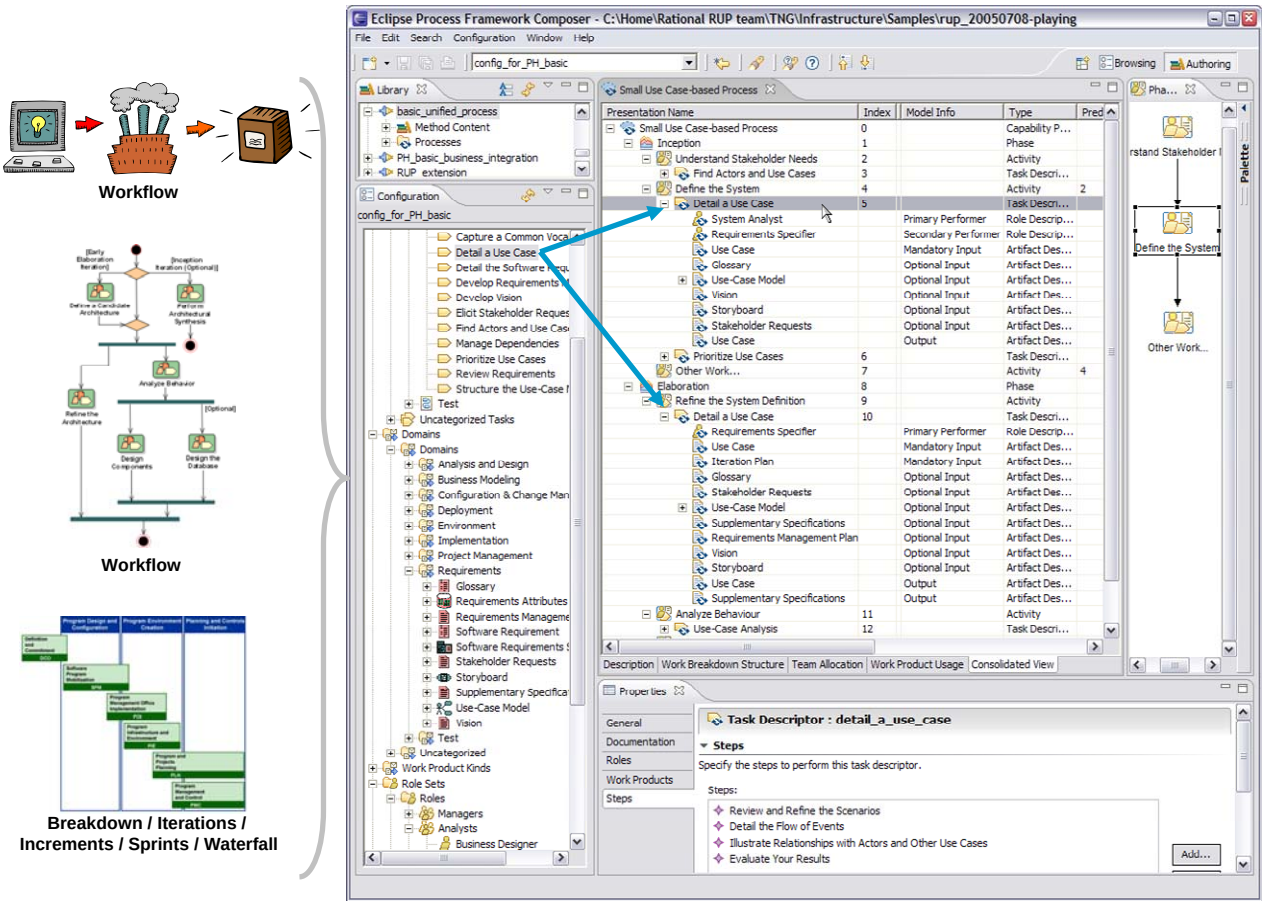

Figure 5: Processes are expressed as workflows or breakdown structures in Eclipse Process Framework Composer. They utilize method content and define how methods are being applied throughout the project lifecycle they describe.

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.[15] 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.

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

The Eclipse Process Framework aims at supporting processes based on many different development approaches, development culture, and development process representation. EPF Composer can be used to define different lifecycle models such as waterfall, incremental, or iterative lifecycles. EPF Composer also supports different presentations for process such as work-breakdown structure or workflow presentations.

The EPF Composer 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 of the activities: the inception phase. The figure also illustrates how method content such as tasks that define which role is performing work with inputs and outputs can be utilized in processes. It indicates 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 secondly, in the elaboration phase in the activity "Refine the system definition". Below each of these task applications 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 applications, expressing differences in performing the detailing use cases method 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. As you will see in Part Two of this article, the application of tasks into processes is a purely optional thing, which helps to make processes express explicitly how they shall be performed; or at least proving guidance on how it could be performed. However, processes in EPF Composer can also be kept free of this explicit information and be kept to expressing minimal or no prescriptive task.

2.4 Summary: EPF Concepts to Create Method Frameworks

Figure 6 shows how the key Eclipse Process Framework concepts relate to method content versus process separation and how you bring instances of these concepts together to build a Process Framework. Typically the word framework is used in the context of an environment one can use to define partial or complete solutions for a particular problem domain using reusable structures as starting points as well as building blocks for creating these new solutions. A Process Framework is a similar thing for the methodology domain: It defines reusable method content, processes, building blocks, and tools that one can utilize to define project or organization specific processes and methods. The EPF Website currently provides three separate process frameworks for user to choose from (such as the OpenUP/Basic, Extreme Programming, and Scrum frameworks[16]), but you can also use EPF Composer to create complete new frameworks from scratch.

[16] See http://www.eclipse.org/epf/downloads/downloads.php.

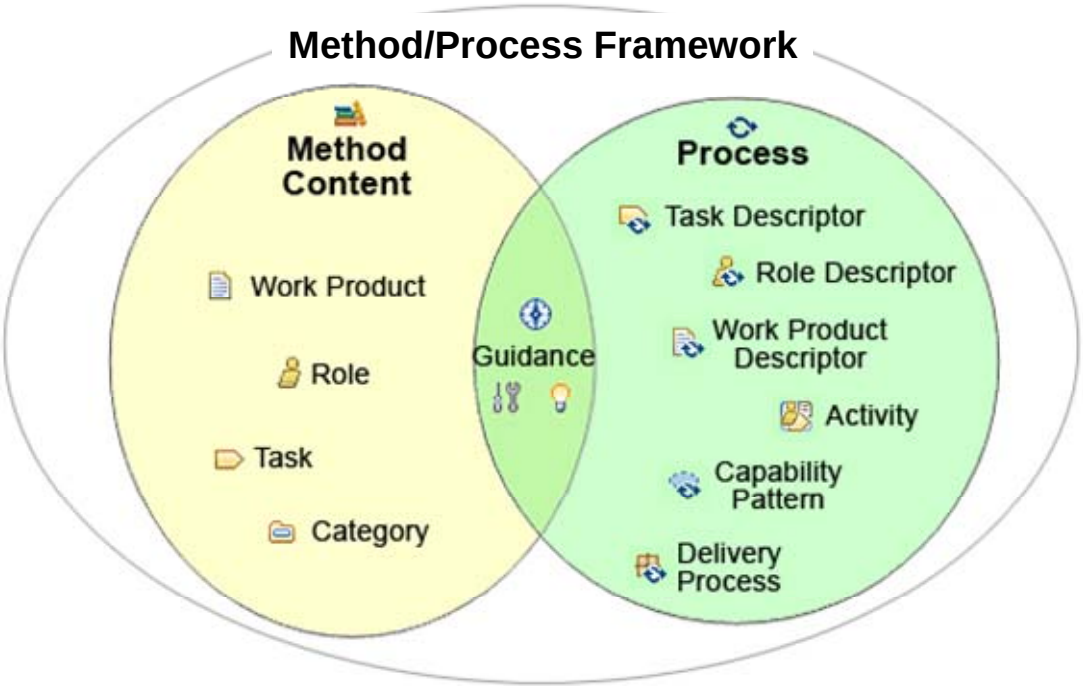

Figure 6: A summary of the proposed terminology overview for the Eclipse Process Framework.

As you see in Figure 6, method content is primarily expressed using work products, roles, tasks, and guidance. Guidance, such as guidelines, concepts, templates, checklists, examples, or roadmaps, is positioned right in the intersection of Method Content and Process. In other words, guidance can be related to and express information relevant to method content as well as processes. On the right-hand side of Figure 6 you see the concepts used to represent processes in the Eclipse Process Framework. 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 references to method content, which we are going to introduce as descriptors in Part Two of this article. Activities are used to define processes of which the Eclipse Process Framework suggests 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 can be 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.

The concepts depicted in Figure 6 are defined in a way that they can be used quite independently from each other, which allows you to use EPF Composer to define process frameworks for different audiences with relatively different process needs, process cultures, or even a different understanding of what process constitutes. In other words, you could define your process framework by using all of the concepts or just the subset that fits your needs. For example, if you work in a very agile environment you might just create descriptions of your guiding principles, practices, and values as guidance elements. If you want to add a bit more formability then you could add definitions for standard roles and work products. The next level of formability would add explicit descriptions of task defining how standard development work shall be performed. All people would work with at that level would be this collection of tasks that can be used just for educating new team members, but also for creating concrete work items in a team support or scheduling system. All of these concepts could be used and managed in EPF Composer as described above without defining and using any processes, thus, by just providing a knowledge base of practices.

Processes can be used in a similar independent way without using any method content at all. For example, teams that inherently know their practices and do not need any additional documentation could just use the process concepts to outline the lifecycle of their general development projects defining key milestones and high-level activity on how to achieve these. As we will show with concrete examples in Part Two of this article, a light-weight process might contain only of a definition of phases or iterations, the milestone for each iteration as well as a list of key deliverables that shall be produced for each iteration, and perhaps lists of roles being responsible for these deliverables. Any combination of both of these sides (the method content only approach or the process only approach) is also possible: You can define processes in EPF Composer that either use a full set or just use a minimal set of method content. Finally, if you work in a very rigor, compliance focused environment you may choose to define rich processes that

integrate with explicit method content, provide detailed guidance as well as categorization schemes that let you clearly define practices and link compliance criteria and checklists for each of these.

3 Typical Usage Scenarios for Working with EPF Composer

If you are working with process framework comprising of method content and processes from the EPF project such as the OpenUP/Basic framework, then you are mostly interested in using and tailoring this content. Thus, this section provides an overview to the most typical usage scenarios for the Eclipse Process Framework Composer for using and tailoring an existing process framework. In Part Two of this article we will extend on these scenarios with additional examples that you would consider when developing new process frameworks with EPF Composer. Although EPF Composer is a powerful process authoring environment, there are various degrees of authoring and modes of utilizing EPF Composer's capabilities. As mentioned before, EPF provides content for various development approaches in its OpenUP/Basic, Scrum, and Extreme Programming frameworks comprising of a lot of method and processes content for different needs. 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 EPF Composer involves simply using the processes the way they ship. For the EPF Composer you are able to obtain rapidly growing libraries of content donated or sold by the project's committers. The commercial version of EPF Composer called IBM Rational Method Composer for example, 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 J2EE or different development circumstances such as adopting a commercial of the shelves system (COTS).

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 EPF Composer's authoring capabilities provide you with easy-to use but powerful editors to do this in a seamless way. The following subsections summarize the most common usage scenarios for EPF.

3.1 Selecting and configuring existing method content and processes

This represents one of the simplest EPF Composer usage scenarios. You select the process and underlying method content that fits your needs by browsing the method library, which contains all of your purchased content as well as content that you downloaded from the Eclipse Process Framework Web site. Once you find a close fit, you start configuring this process (as depicted in Figure 7) by selecting or deselecting what EPF calls "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" J2EE versus J2EE for SOA-specific content).

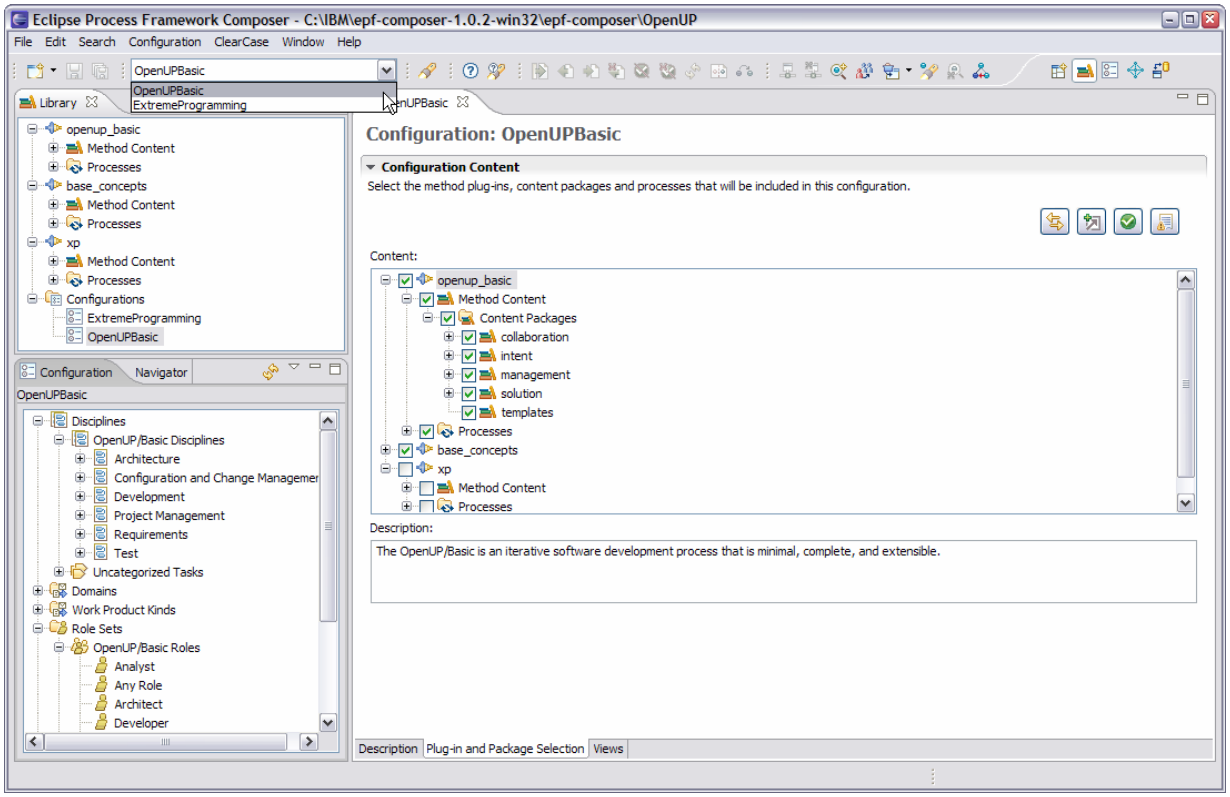

Figure 7: Configure processes and method content by selecting or de-selecting method packages that shall be referenced or not referenced by the processes. You can create several method configurations defining filters on the overall method library in EPF Composer. Use the combo box in the toolbar to quickly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

Method content and processes are organized in EPF method libraries 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 sub-packages for specific practices in these disciplines. For example, OpenUP/Basic factors all of its content related to visual modeling into a separate package. Thus, you can add or remove visual modeling from the OpenUP/Basic with just one simple mouse-click by selecting or de-selecting the right 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.

3.2 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 the EPF Composer editors. You can create what we call a process contribution for an existing process that defines differential changes to that process. You can also 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. EPF Composer 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 loosing your changes. (In Part Two, I will provide detailed examples for process tailoring.)

3.3 Create a new process

As an alternative to tailoring an existing process, you can also author a completely new process that reuses activities 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 as well. 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 in a common process area that are performed typically over and over again. Examples of capability patterns include "manage use case model" “develop a component,” “validate build,” or "ongoing management and support." In addition to replicating such patterns each time the work described by the pattern will be performed, EPF Composer allows you to dynamically link patterns into your process. When the pattern changes, all applications of the pattern will then be automatically updated.

EPF Composer's process authoring capabilities let you fully utilize the separation of method content and process as outlined in Section 2. 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 demonstrating how method content and capability patterns have been re-used in two processes with different lifecycle models.

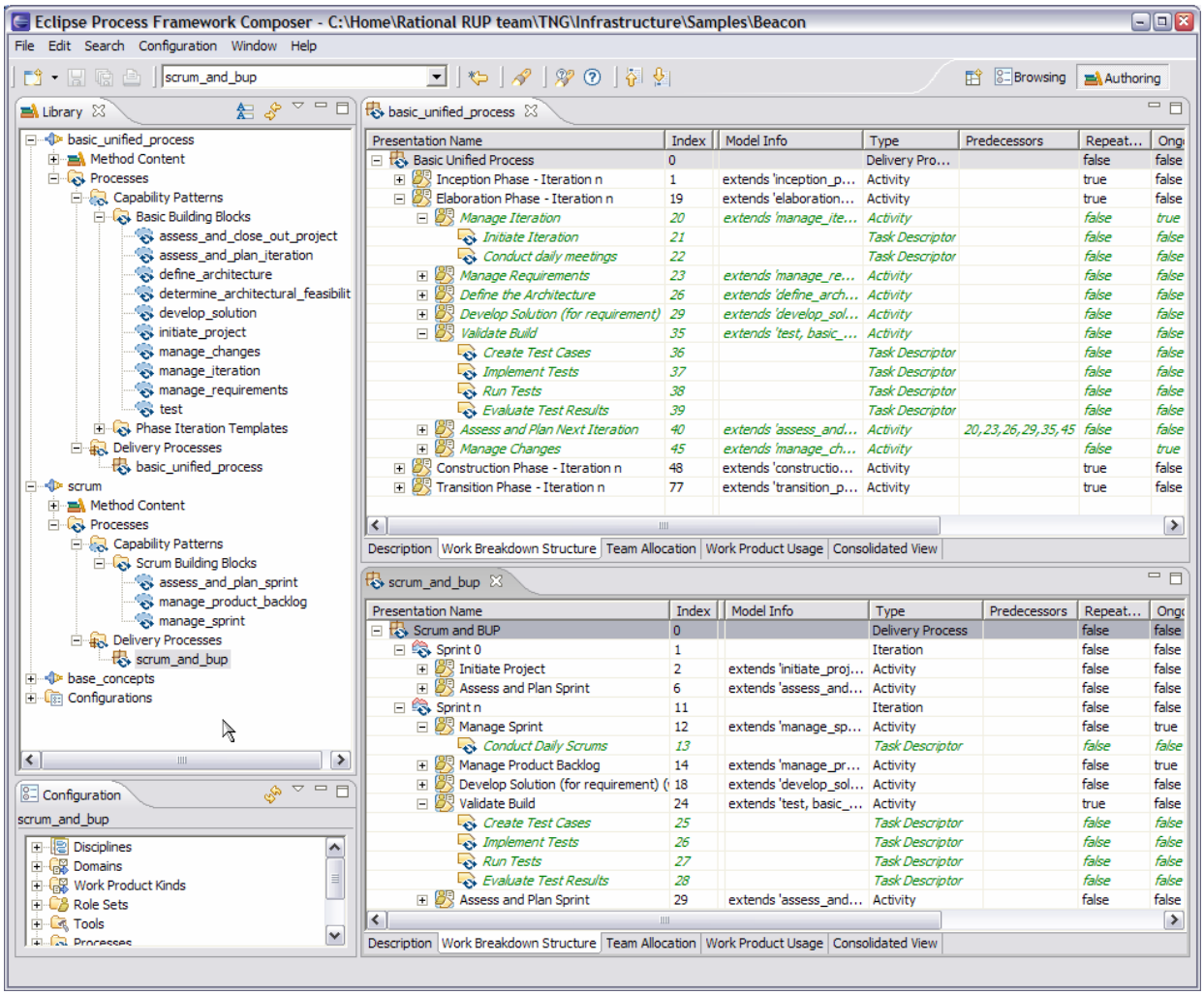

Figure 8: Two processes with two different lifecycle models applying the same capability patterns and method content.

In Figure 8, the right hand side contains two breakdown structure process editors. The top one defines a simple variant of the Unified Process, which was the precursor to OpenUP/Basic. The bottom one defines

a Scrum-like process. Each of the two has its own distinct lifecycle model. The top one uses the four UP 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 building blocks for the upper 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 EPF Composer's process authoring capabilities by just reusing patterns and tasks that could serve as guidance for learning rather than strict work instructions for perhaps less experienced teams.

3.4 Develop method content and create or extend processes

The last scenario presented here is the ability to not only create or tailor processes reusing EPF or a 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 EPF Composer's unique method plug-in and variability capabilities. Variability allows you change existing content without directly modifying the original. You have, for example, the ability to create a method plug-in for EPF's OpenUP/Basic that contains a task contribution adding some new steps to an existing task. Or, you could define a replacing work product for an OpenUP/Basic 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 OpenUP/Basic 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 reapplying your changes.

In EPF Composer, 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. EPF Composer 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).

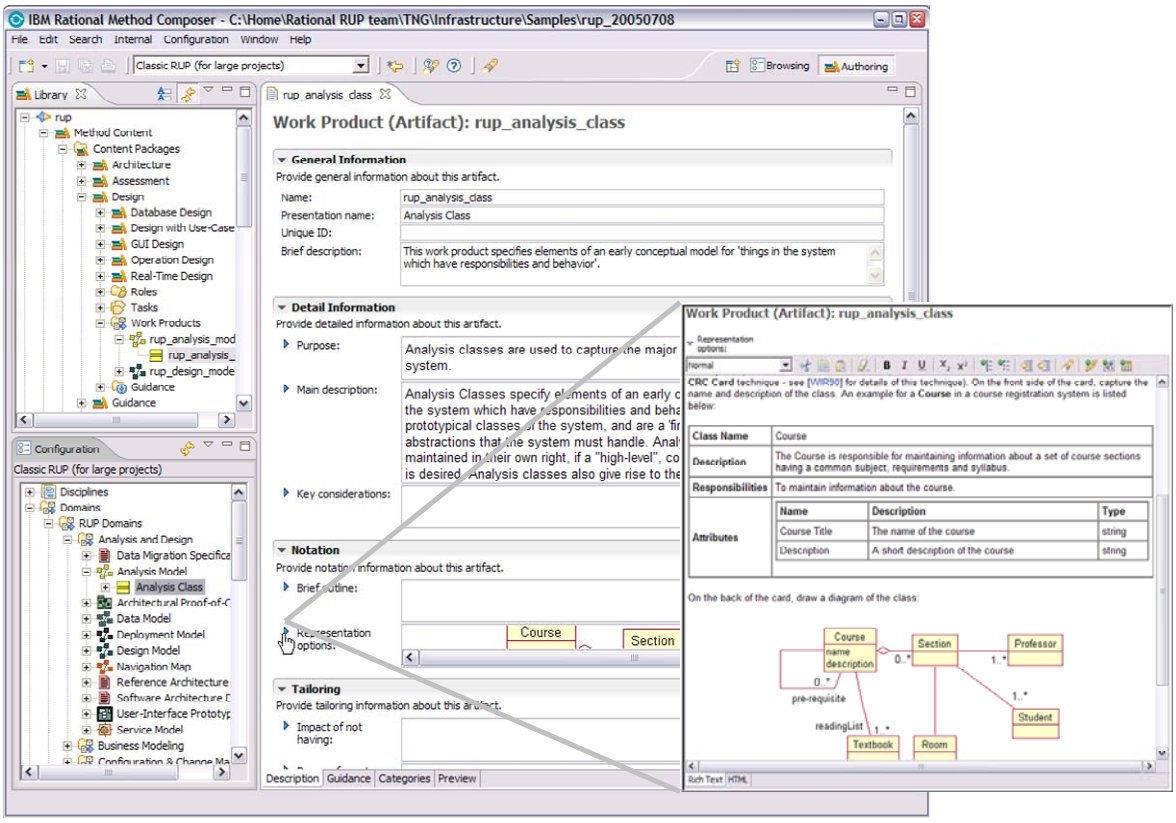

Figure 9: Form-based editor for a work product. Every form field can be expanded into a full-sized rich text editor that allows you to format text, work with tables, include images, and so on.

Once you have created your own method content elements, you can include them into 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 EPF Composer usage scenarios. In Part Two of this article, I will present a more in-depth look at method content and process.