

Increasing Development Knowledge with EPFC

Are all your developers on the same page? Are they all using the best practices—and the same best practices—for agile, iterative, architecture-centric, risk-driven and quality-driven development? Maybe, maybe not: Even when team members are working on the same project, they may not have access to the same information or the same version of the information. It's difficult to combine and integrate content and development processes that are made available in their own proprietary format. It is even more difficult to define an organized and systematic development approach that addresses an organization's specific culture, standardized practices and compliance requirements.

That's where the Eclipse Process Framework comes in. The EPF project provides an extensible framework and tools for defining and managing software development methods. Within this framework, the project develops content for a range of software development and management processes that are applicable to a broad set of development platforms and applications. Whether this information is relevant to an individual organization or the entire Eclipse community, it can reduce the duplication of development and research efforts.

THE EPF APPROACH

The Eclipse Process Framework Composer is a process-management tool platform and conceptual framework for authoring, tailoring, documenting and deploying development process frameworks in various formats. For example, EPF Composer can publish HTML content and exported project plans. It was designed for engineers, project leads and project and program managers who are responsible for maintaining and implementing processes for development organizations or individual projects.

EPF Composer serves two main purposes, as shown in Figure 1. Its first goal was to provide a standardized representation and managed libraries of reusable method content that developers need to understand the methods and key practices of software development. These basic practices include eliciting and managing

■ BY PETER HAUMER

requirements, methods for analysis and design, implementing a design and test case, testing against requirements and managing the project scope.

The tool helps development professionals set up a knowledge base of intellectual capital that lets them browse, manage and deploy content. This content can be licensed, acquired, or—perhaps most importantly—developed in house. It can comprise method definitions, whitepapers, guidelines, templates, principles, best practices, internal procedures and regulations, training materi-

EPF PROVIDES AN EXTENSIVE FRAMEWORK FOR DEFINING AND MANAGING SOFTWARE DEVELOPMENT METHODS.

al and any other general descriptions of how they want and need to develop software. The information can be used for reference and education and as the basis for developing standard processes. All managed content can then be published to HTML and deployed to Web servers for distributed use.

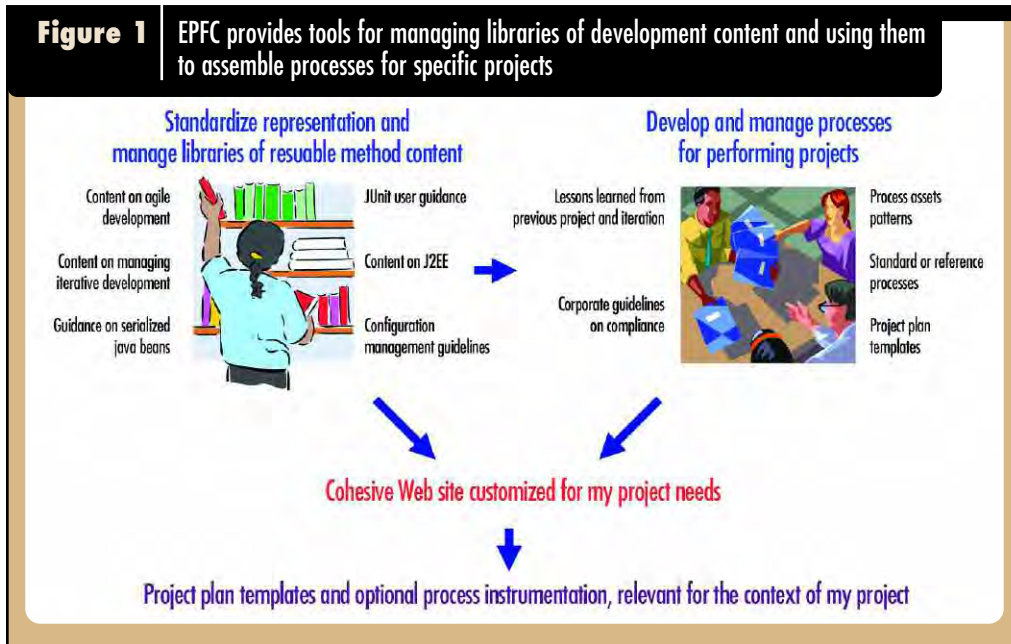
The second goal of EPF Composer is to support the systematic growth and management of development processes. Development teams need to define how to apply their development methods and best practices throughout a project lifecycle. Management methods have to be applied in one way during the early phases of a project, where the focus is more on stakeholder needs and requirements. The same methods have to be performed in a different way dur-

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

ing later phases, where the focus is on managing requirements updates, changes and the impact of these changes.

Teams also need a clear definition of how the different tasks within the methods relate to each other. Change management methods impact the requirements management method and regression. EPF Composer provides process-engineering capabilities by supporting engineers and project managers in selecting, tailoring and rapidly assembling processes for real-world projects.

The end result should be a catalog of predefined processes for typical situations that can be adapted to multiple projects so that organizations can develop their own libraries of processes. The process building blocks are called capability patterns and represent best development practices for specific disciplines, technologies or development styles and form a toolkit for quickly assembling these processes. Additionally, the documented processes created with EPF Composer can be published and deployed as Web sites or files.



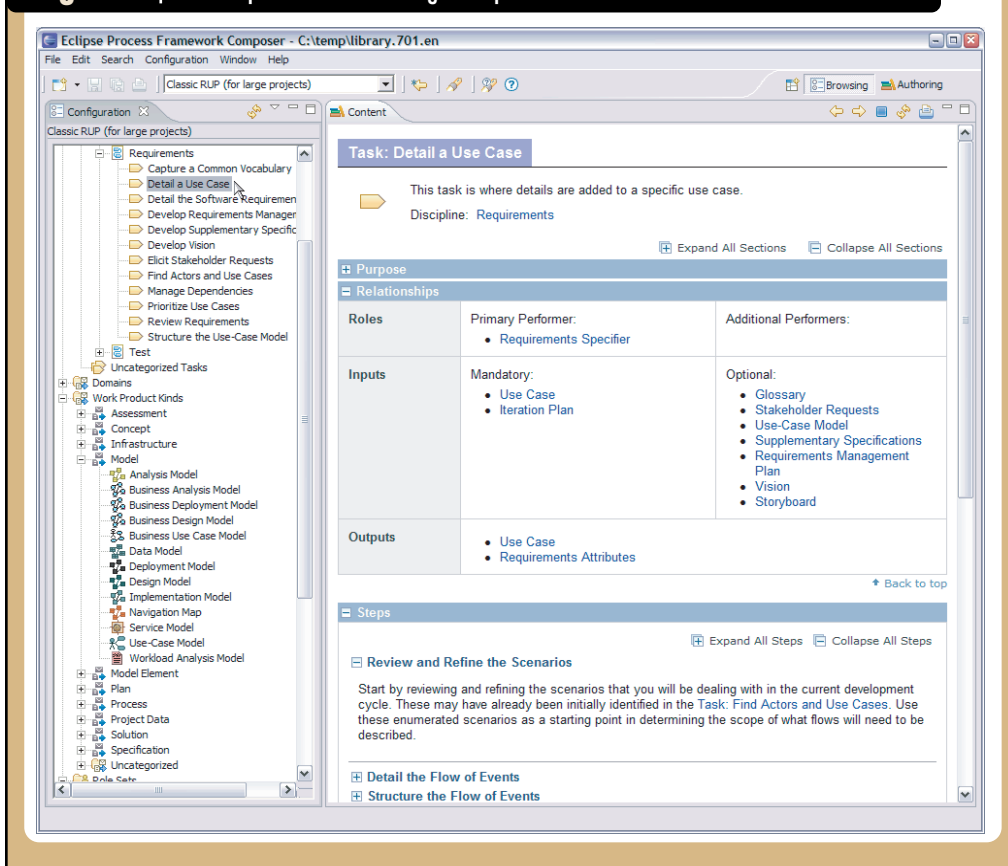
REUSABLE CONTENT LIBRARIES

To learn about development methods, most people do research in libraries or they receive training and mentoring. Many development methods are typically described in books, articles, training materials, standards, regulations and other forms of documentation. The sources often document methods in different ways, but they usually provide step-by-step explanations for a particular means of achieving a goal under general circumstances. For example,

developers might learn how to transform a requirements document into an analysis model, define an architectural mechanism based on functional and non-functional requirements, create a project plan for a development iteration or define a quality assurance plan for functional requirements.

EPF Composer allows you to take this content and structure it in a specific way using a predefined schema. This schema is an evolution of the OMG's Software Process Engineering Metamodel 1.1 specification (www.omg.org/spem) specification. SPEM helps you organize large amounts of data for development methods and processes. Following this plan, method content is represented as a construct of roles defining development skills and responsibility.

Figure 2 | EPFC expresses content using concepts such as tasks and roles



ties for work products that are produced by tasks and have work products as inputs and outputs.

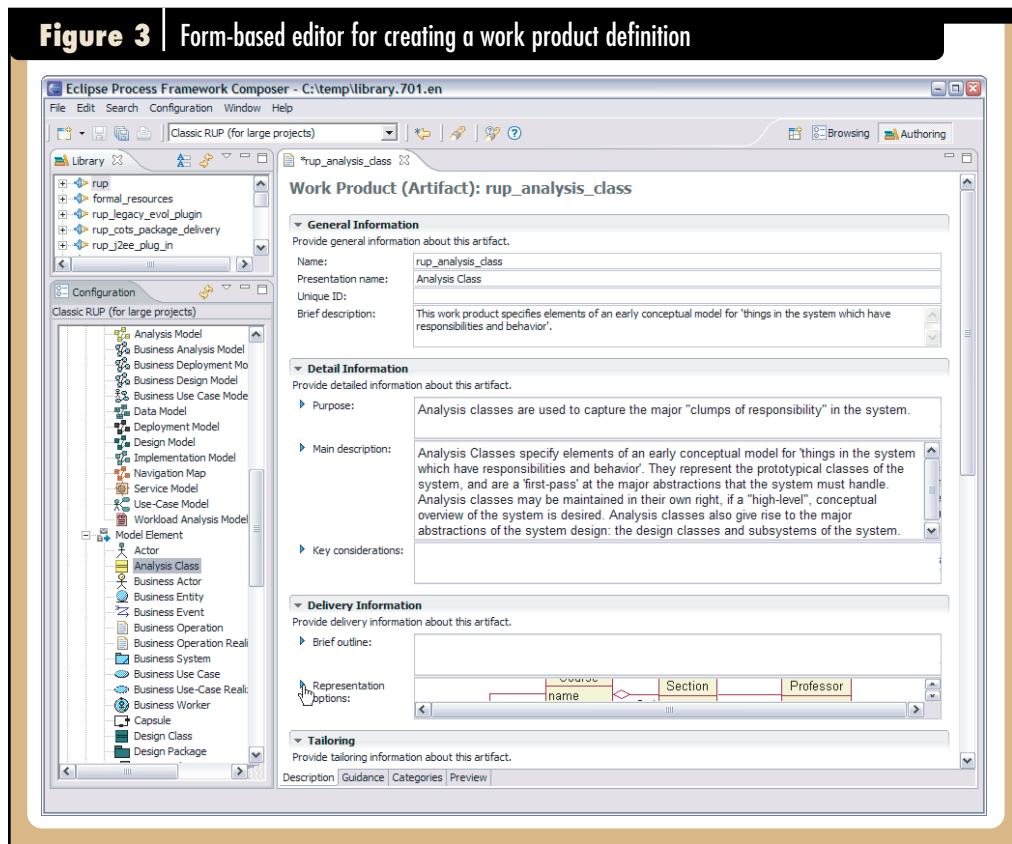
Figure 2 shows how method content is represented. Method content elements can be organized in tree browsers like the one on the left to provide different indices of the available elements for rapid access. The task presentation is an example of how the information

capabilities for the creation of tables, images, hyperlinks and formatted text. These editors manage HTML that can be directly manipulated from another view within the same editor. In addition to managing the method elements, roles, tasks and work products, you can add specific guidance elements that can contain supplementary and background information. For example, you can add supplementary free-form docu-

mentation such as whitepapers, concept descriptions and guidelines. You can also add document templates and review checklists for work products, roadmaps, concrete examples and reusable assets to supplement these learning materials.

CREATING DEVELOPMENT PROCESSES

A development process defines how work will be performed and how products are produced and evolve. In other words, development processes define how development projects are executed. Processes define how to get from one milestone to the next by defining sequences of work, operations and events that use resources such as time and



can be rendered into HTML for deployment. It shows a use case in terms of steps that need to be performed to achieve the task. The task has various relationships, such as an association with the Roles Requirements Specifier.

The Detail a Use Case task defines the products that are required to perform the work (inputs) as well as the work products produced or updated (outputs). EPF Composer maintains and presents relationships between roles, work products and tasks as hyperlinks connecting these elements to each other. To help you create and manage information for your own needs, it provides you with several content-management capabilities, such as the form-based editors shown in Figure 3, which depicts the editor for a work product definition.

Using these forms, you can add your own method content, tailor the content that is being developed in the EPF project for your needs or make use of content that will be provided for licensing from vendors or other organizations. The editors allow you to “model” the relationships between elements. They also provide HTML-based editing features.

As shown in Figure 4, every form field can be expanded into a full-sized editor that provides rich-text editing

expertise and produce a predetermined outcome.

Defining a strict sequence—as in a waterfall model—is no less a process than defining a semi-ordered sequence of iterations for parallel work. They simply represent different development approaches. Whereas traditional waterfall-based processes are able to define more or less straight lines of sequences for the tasks to be performed, modern iterative development processes, such as RUP, DSDM, OPEN, XP and Scrum require more complex structures to represent increments of development. In these structures, work is packaged into repeatable pieces in which tasks are performed concurrently and repeatedly.

All of these development approaches require different degrees of formality and levels of detail for process descriptions. Traditional development processes define phases based on predefined deliverable types, providing exact descriptions of the sequences of work that produce these deliverables.

Agile development processes only ask for informal work descriptions because they assume that teams are self-organizing and are able to perform the work without guidance and without control once the development goals have been defined. They also focus on describing work

sequences based less on deliverables and more on capability (result-based) milestones. Modern scalable iterative processes provide explicit descriptions of the work to be performed with a variable and flexible amount of detail depending on the type of project and maturity of the organization.

The EPF project provides one unified process framework that supports the management of processes from all of these approaches. In other words, the Eclipse Process Framework will support processes from many different development approaches including development culture and development process representation. EPF Composer can be used to define waterfall, incremental, iterative and other lifecycle models. It also supports progress representations that include work-breakdown structures and workflow presentations. Figure 5 shows how processes are created and presented with EPF Composer's process editors. It shows the same process in two consistently maintained presentation formats: as a breakdown structure and an activity diagram.

Many iterative development processes are modeled using nested workflow representations to support complex process structures as well as various degrees of formality. EPF Composer's activity diagram editor, which you'll see at the lower half of Figure 5, allows you to define complex process structures as a hierarchy of nested diagrams in which each diagram node could be refined or detailed with a whole new diagram. Activity diagrams can express parallelism using fork, merge and join nodes. They can also show sequencing using control flows.

Another popular presentation for traditional waterfall as well as iterative development processes is the work breakdown structure that you'll see in the top half of Figure 5. Breakdown structures are similar to nested activity graphs because they also allow the refinement of work definitions via nesting. They provide an easy way to sequence work using what are called predecessor dependencies, which are similar to control flows. All elements that are not related via these predecessors are free to be performed in parallel.

Another advantage of the breakdown structure presentations is that they are very popular in project enactment and management applications such as IBM Rational Portfolio

Manager or Microsoft Project. Using breakdown structures in EPF Composer can bridge the gap between process definition and process enactment. It melds these two popular process representations into one format that it maintains internally to represent processes. Based on this format, RMC supports both presentations inter-

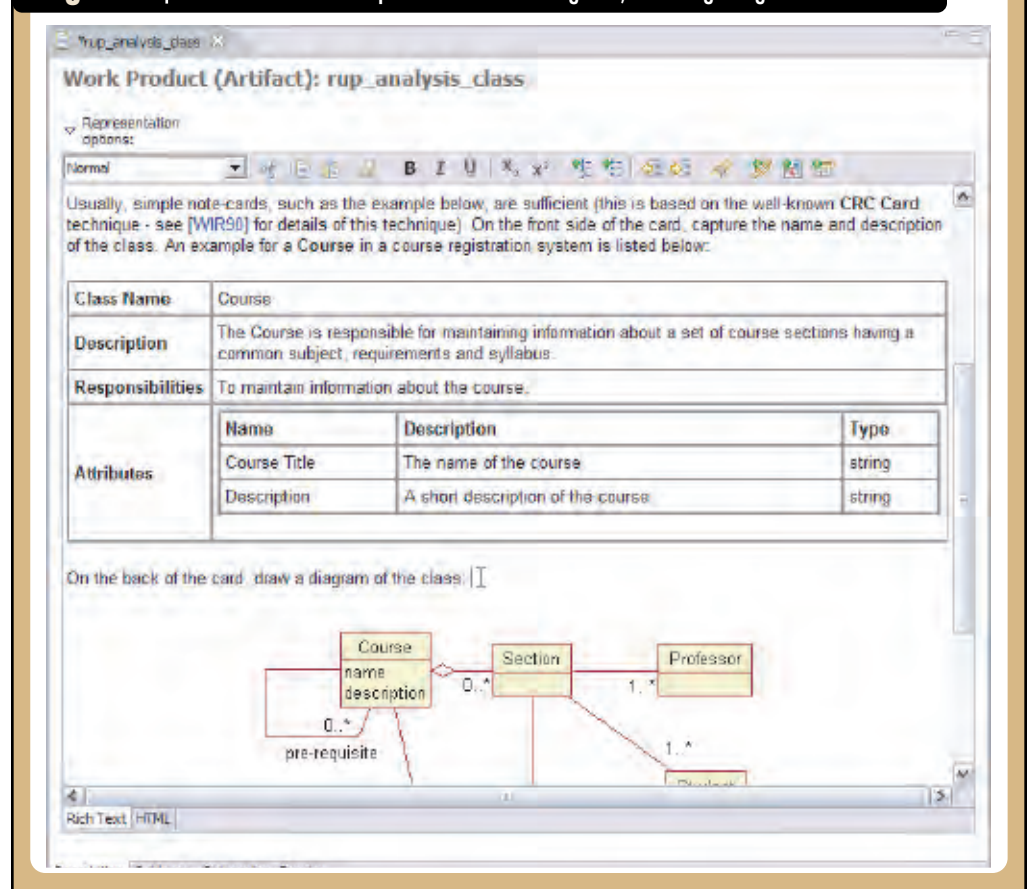
MANY MODERN SCALABLE PROCESSES PROVIDE DESCRIPTIONS OF THE WORK TO BE PERFORMED.

changeably, as you can also see in Figure 5.

Last but not least, Figure 6 shows you how method content relates to processes in EPF. In defining a process, you can take method content and systematically combine it into specific process structures that specify how the work described in your method content will be organized over time following a specific development approach. You apply and tailor the content to meet the concrete needs of a particular type of development project, such as software for an online system or software and hardware for an embedded system.

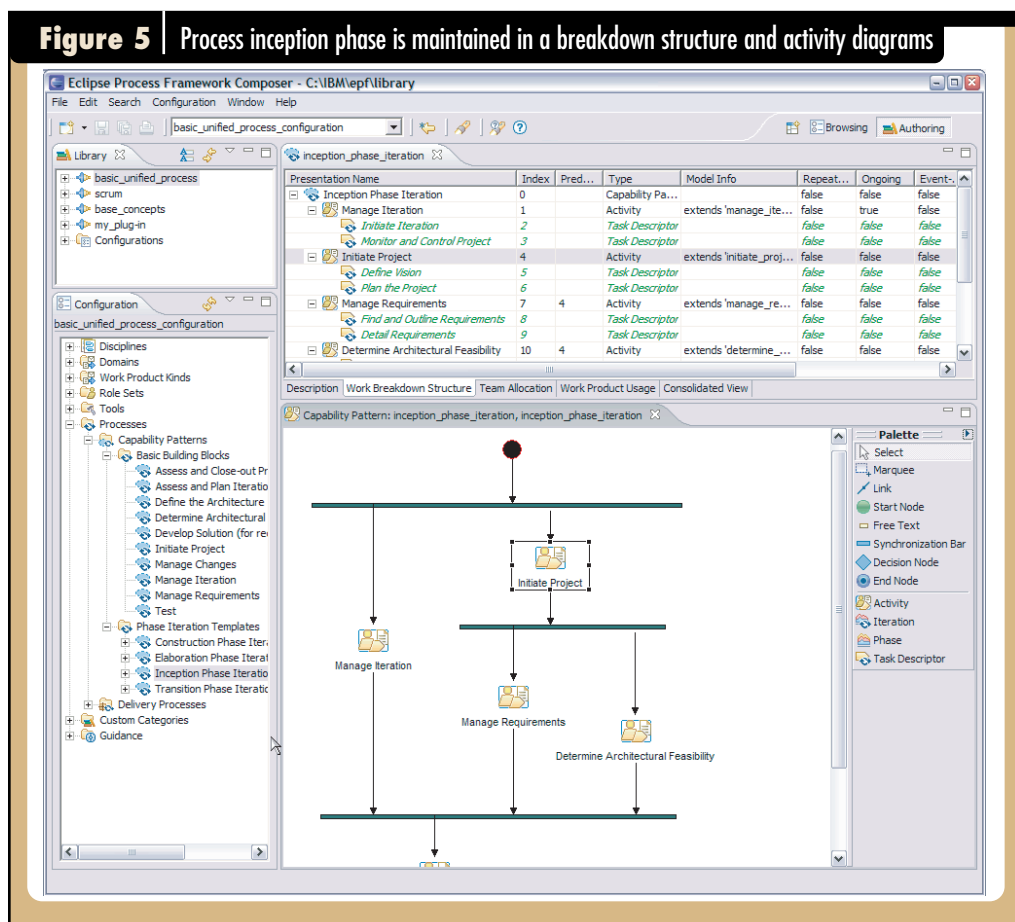
The two blue arrows in Figure 6 show how the Detail a Use Case task (from the method content library of Figure 2) has been applied in the same process twice—first in the inception phase under “Define the System”

Figure 4 | Form fields can be expanded for formatting text, including images



and then in the elaboration phase in “Refine the system definition.” Below each of these task applications you’ll see lists of the performing roles as well as the input and output work products. If you look closely, you’ll see that these lists are different for each of these two task applications, expressing differences in performing the use cases throughout the lifecycle.

Here, the process author tailored the tasks that were reused from the content library. These tasks were then used to model phase-specific variances of performing the work for the exact focus of the task performance in each point of the lifecycle. You can also see the different roles and changes in the list of inputs to be considered as well as the outputs that will be produced or updated.



In addition to updating the roles, input and output work products for a task, you can also provide additional descriptions and select the exact steps of the task that should or should not be performed in this particular occurrence of the referenced task.

Once you have defined and documented your process models in EPF Composer, you can make them available to the development teams in two different ways. The first is to publish them as Web sites that provide hyper-linked navigation and browsing capabilities along the breakdown structure and activity diagrams. Additionally, they can be exported into a template that integrates with a project management tool to provide links from the templates to the published Web sites that contain the process and method content documentation.

FROM CONCEPTS TO CONTEXT

Figure 7 summarizes the key EPF concepts for creating process frameworks. A process framework defines reusable method content, processes, building blocks and tools that you can use to define project or organization-specific processes and methods. The EPF project will develop several process frameworks that address different audiences. These can be deployed with EPF Composer “out of the box.” You can also use EPF Composer’s authoring capabilities to create complete new frameworks from scratch.

As we have seen, method content is primarily expressed using work products, roles, tasks and guidance. Guidance is positioned right at the intersection of Method Content and Process, because it can express information relevant to method content as well as processes. On the right-hand side of Figure 7, you’ll see the concepts used to represent processes. The main concept is the activity that can be nested to define the breakdown structures and activity diagrams in Figure 5 and Figure 6. They contain references to method content and are used to define processes. The Framework suggests two types of processes.

Delivery processes represent a complete and integrated process template for performing one specific type of project. They describe an end-to-end project lifecycle and can be used as a reference for running projects with similar characteristics. The framework also suggests capability patterns that express process knowledge

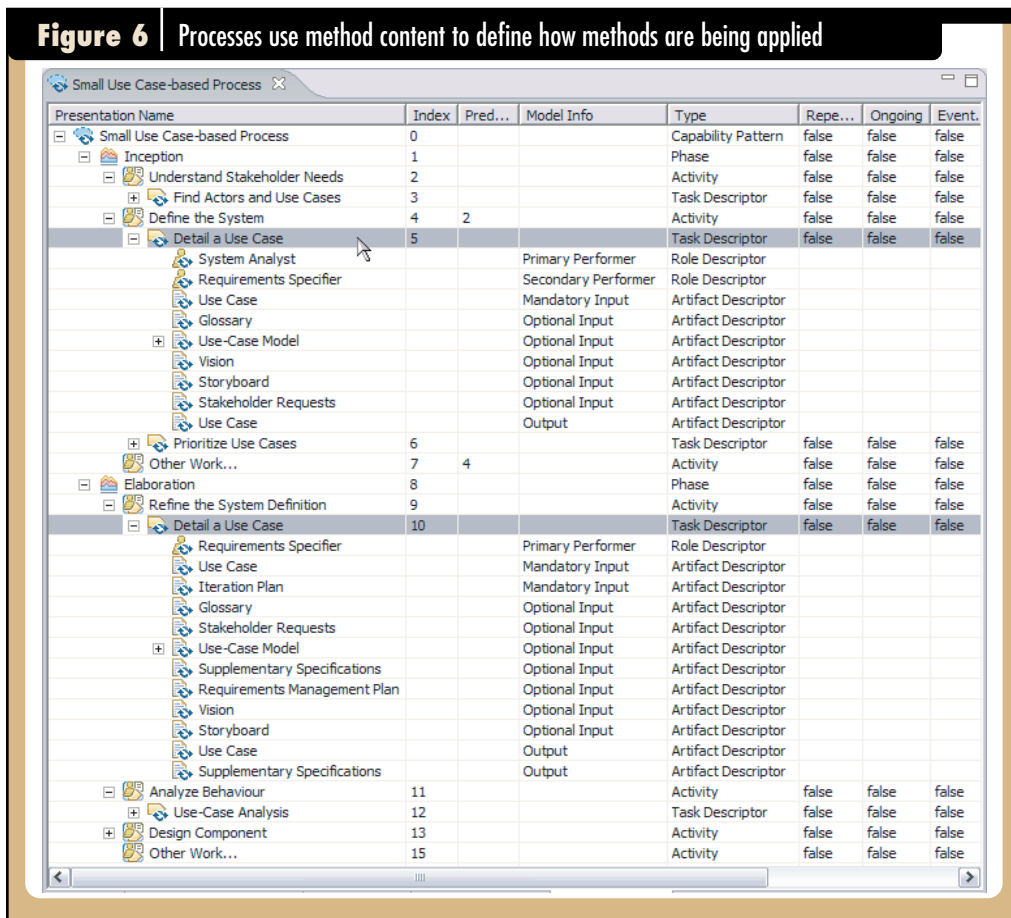
for a key area of interest, such as a discipline or a best practice. They can be directly used by process practitioners as a guide in their phase of development. Capability patterns can also be used as building blocks for assembling delivery processes or larger capability patterns to ensure optimal reuse and application of key practices. EPF Composer provides authoring methods for applying and nesting capability patterns to processes by reference or value.

The concepts shown in Figure 7 are defined to be used independently of one another, which lets you use Composer to define process frameworks for different audiences that have different needs, cultures or even a different understanding of what constitutes a process. In other words, you could define your process framework

by using all of the concepts or just the subset that suits 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. To add a bit more flexibility, you could add definitions for standard roles and work products. The next level could add explicit descriptions of tasks, defining how standard development work should be performed. This collection of tasks could be used for educating new team members or for creating concrete work items in a team support or scheduling system. All of these concepts could be used and managed in EPF Composer to provide a knowledge base of practices—without defining and using any processes.

Processes can be used in a similar independent way without using any method content at all. For example, teams that 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. A light-weight process could contain phase definitions, milestones for each step, a list of key deliver-




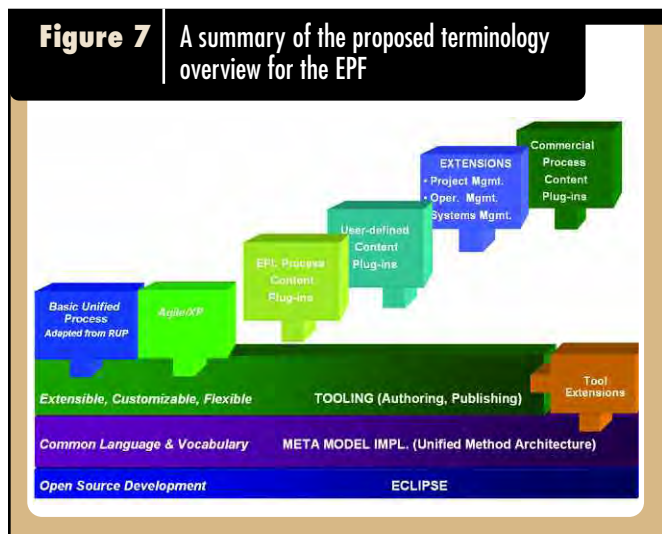
processes that integrate with explicit method content. In this type of environment, you might need to provide detailed guidance and categorization schemes that let you clearly define practices and link compliance criteria and checklists.

AN ENCYCLOPEDIA IN THE WORKS

The Eclipse Process Framework is an open-source project in itself. Contributors are invited to participate at any level to add reference materials and tools relating to any Eclipse process. The project uses the Eclipse Modeling Framework to define and manage the Meta Model implementation illustrated in Figure 7. The EPF Composer user interface is built on top of this layer using Eclipse’s Rich Client Platform tools.

EPF Contributors can extend this existing user interface with new capabilities or develop complete new user interfaces, such as Web-based authoring and collaboration tools. Finally, the EPF project develops method content and processes for a range of software development and management processes applicable to a broad set of development platforms and applications.

Contributors are invited to join the project’s content development committers in this work or develop new additional content. The Eclipse environment is the result of collaboration. By creating information and sharing processes that may be applicable to others in the community, you reap the rewards of collaboration in a whole new way, streamlining your own development and helping others to do the same. 



ables and lists of the developer roles responsible for them. On the other hand, if you work in a very rigorous, compliance-focused environment, you can define rich

A BZ Media Publication

ECLIPSE review

Volume 1 ■ Number 2
Spring 2006 ■ \$8.95

www.eclipsereview.com



**Callisto:
10 Eclipse P
Hit the S**

**Gnireenigne F
Reverse F**

Ri

**n
Eclipse**

Subscribe Today!
www.EclipseReview.com