JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2010**

Vol. 9, No. 2, March - April 2010

Attached Processes

John D. McGregor, Clemson University and Luminary Software LLC, U.S.A.

Abstract

Many software engineers try very hard not to use the word “reuse” because many methods and tools intended to promote reuse have failed to meet expectations. As soon as they learn that a software product line involves reuse they turn away quickly. The “attached process” that accompanies every core asset in a product line makes software reuse effective instead of frustrating. In this issue of Strategic Software Engineering I will describe how attached processes are created and how they fit into the core asset and product building processes.

1 INTRODUCTION

There are many jokes about consumers as product builders. The parent trying desperately to put together the bicycle after the children go to bed on Christmas Eve. The newly wed trying to assemble furniture from a kit. Or, the user’s manual written in stilted prose. In each case the instructions never quite match the materials the product builder has in front of them. Another of the difficulties with software (re)use is that the user often tries to understand the software completely rather than just understanding how to use the software. I will discuss how to structure the information about using an asset in ways that are as useful as possible and as easy to digest as possible.

Quite often when I speak to a group about software product lines I am faced with a skeptic in the audience. They have participated in projects that made promises about faster time to market because the project intended to reuse legacy code as a base upon which to build a product. When I talk about a core asset being a reusable asset, they moan. By the time I finish explaining about attached processes as users’ manuals the light bulb has come on for many people. They get it. Others at least acknowledge that it is better than the typical documentation accompanying a piece of software.

Every asset intended to be used in multiple products, a **core** asset, is accompanied by an attached process. The attached process is essentially a user’s manual for the asset. Whether the asset is a piece of software or a tool, or a document, the attached process provides useful information to those using the asset to build products. In my opinion the attached processes for the core assets are as important to the success of the software product line strategy as is variation management or the product line architecture. In fact both of those elements play a role in the attached processes.

The attached process provides meta-data about the asset to which it is attached. This information describes the **context** in which the asset is used [McGregor 05]. Context is the set of assumptions that influence decisions. The context for a software core asset is the role of the asset as defined in the software architecture. The attached process for a software asset includes references to the attached processes for those assets with which this asset has a "requires" relationship. The context for a non-software core asset is the asset's role in the product line production process.

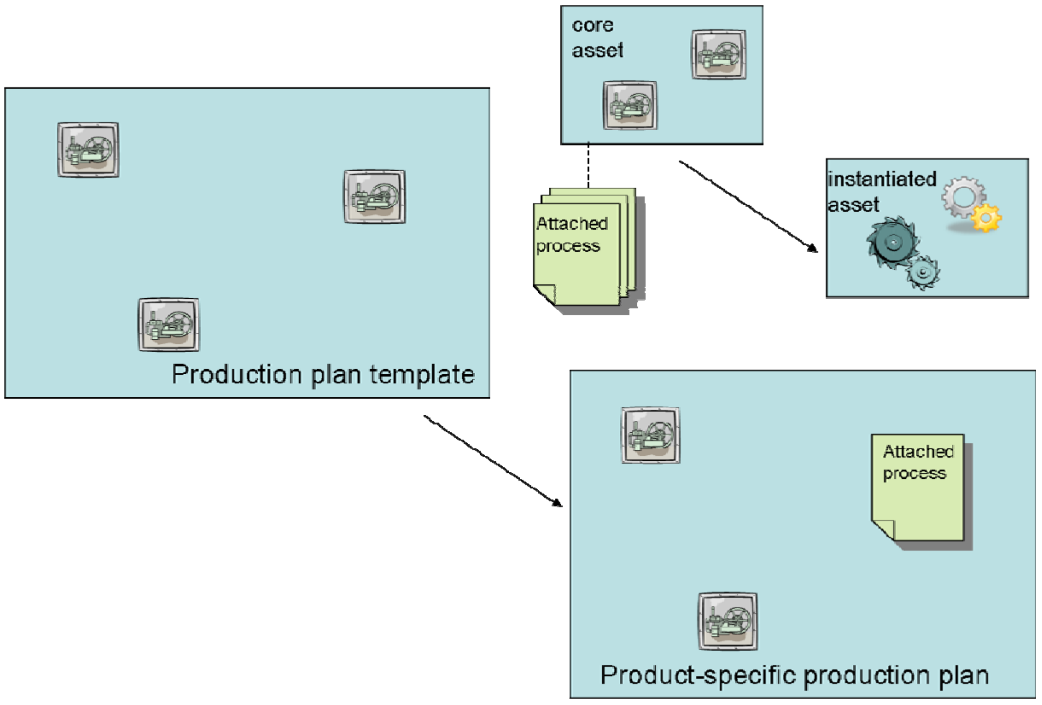


Figure 1 - Role of attached process in instantiation

The attached process includes a configuration guide, which details each variation point in the asset. The guide describes the steps required to select and instantiate the desired variant at each point. The steps, of course, depend on the exact variation mechanism that is used at each point. For example, the attached process may direct the user to create a subclass of a specific base class and to override specific methods to provide required behavior or the user may simply select a value in an enumeration to fill in the variation point.

Figure 1 shows a core asset (in reality there would be many) with two variation points and its attached process and a production plan template with three variation points (although in reality there would be many more). The production plan template is instantiated by resolving its variation points with the attached processes from the selected core assets. Each asset can now be instantiated for the specific product by following the attached process to resolve the variation points.

In this issue of Strategic Software Engineering I will expand on these brief descriptions and discuss some mechanisms for implementing attached processes.

2 CONTEXT FOR THE ATTACHED PROCESSES

The production plan for a software product line describes how to build a product using the core assets. In an earlier column I discussed the production plan in detail [McGregor 06]. The production plan core asset is a template into which the attached processes must fit. As with all core assets the production plan template must be instantiated into the product-specific production plan when it is used for building a product.

The production process for a product is outlined in the production plan template but with missing details. Each variation point in the product is resolved by selecting the variant appropriate for the specific product being built. When the core asset is selected to resolve a variation point in the product, its attached process is used to resolve a variation point in the production process. The resolution results in detail being added to the production process. For example, there might be the choice between deploying the product as a static web page or as a Java EE web application. If the web application is chosen then tools to compress the files into a WAR file are needed and additional steps are added to the production process to apply those tools.

“Core asset” is really a collection of items that are tightly coupled and that are used in a specific context. A core asset has a primary artifact such as a software module or a document, an attached process, a means of evaluating the quality of the artifact, and potentially the definitions of the variants that can be used to resolve the variation points in the asset. These may be software modules, models, or other artifacts. This collection of items is encapsulated so that it can be used in multiple different products in the product line.

3 PLANNING THE ATTACHED PROCESS

The attached process describes how the core asset is used in the production of products. The attached process is written by the creator of the core asset to which it will be attached. Designing the attached process in parallel with the design of the core asset enhances the usability of the core asset and ensures that the attached process gives correct information, describes the complete asset, and provides information that is consistent with the actual asset.

In addition to those items I have already described, the attached process specifies such things as:

- Usual process structure such as who is responsible for each activity in the process and what skills are required
- Effort estimates that can be plugged into the larger schedule

- A bill of materials – what is needed to support the asset
- The order in which the variation points in the core asset should be resolved.
- Configuration settings for tools

As a simple, large-grained example, one familiar variation point in many product lines of embedded systems regards the primary output mechanism for a product. The production plan for the product line would include a step in which that variation point is instantiated. Once the choice is made to use a web browser, the IDE dedicated to that choice would be loaded and further variation points in the production plan could be instantiated. Choosing a different output mechanism would result in a different set of detailed implementation steps being added to the production plan instantiation.

The creator of the production plan template and the creators of the attached processes are jointly responsible for the quality of their integration. The mechanism used to integrate the attached process depends upon the technologies used to define and maintain the production plan. A production plan that is mainly a tool script may simply need text to be inserted at the appropriate point in the script. If the plan is represented by a workflow, new workflow objects may need to be created and added to the workflow.

4 IMPLEMENTING THE ATTACHED PROCESS

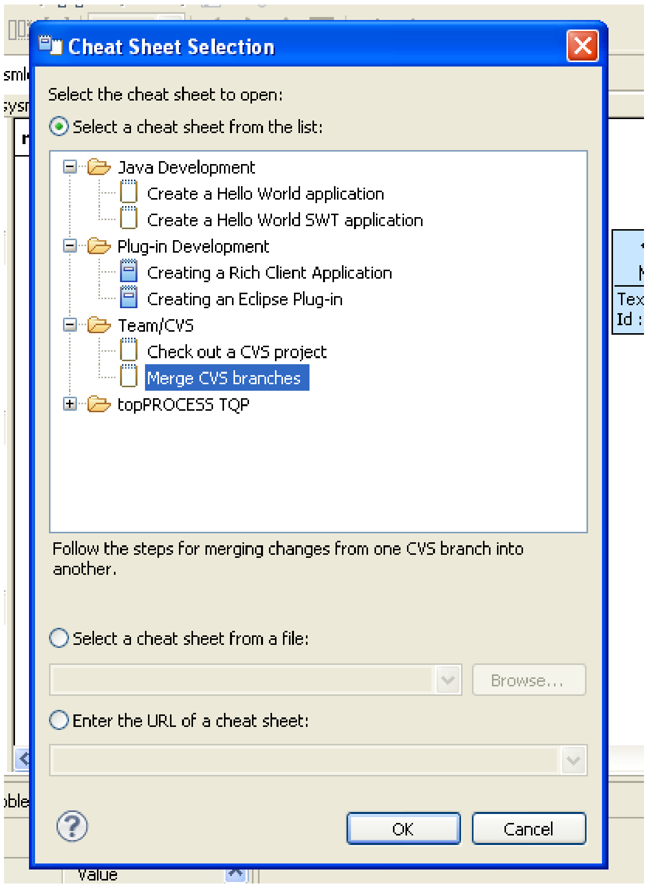
Attached processes are implemented to complement the production strategy of the product line. The techniques used to implement the attached process are chosen with the consideration of who will do the configuration and when. For example, if the user is able to configure the product, the mechanisms defined by the attached process must be available at least at install time if not runtime. The designer of the asset makes the decision about issues such as whether the product must be restarted for the new configuration to be used. The mechanisms to avoid a restart are usually more complex and, in some cases, require an additional level of abstraction that makes execution slower.

One of my colleagues at the SEI says “the truth is in the code”, which might suggest the attached process should include the source code as the best way for someone to understand how to use the asset. The problem with that is the source code is distracting. The product builder spends valuable time thinking of all the ways the asset could have been better implemented. When the attached process defines a straightforward “as-is” use of the asset utilizing pre-defined configuration methods rather than a redesign of the asset, reuse of the asset is more productive.

I want to explore briefly just three technologies for implementing the attached process:

Cheat sheets – The Eclipse framework provides a mechanism referred to as a “cheat sheet.” [Eclipse 09] These are short pages of help that are available through the **Help | Cheat Sheets** menu. The developer of an asset can create dependencies so that when a product builder loads an asset into the Eclipse environment, the cheat sheets related to that asset are automatically listed in the Cheat Sheet menu, Figure 2.

The sheets are structured using an xml tag language. The sheets can contain pointers to extensive resources or can execute actions such as media resources to communicate information about the asset, Figure 3. The xml language can be used to enforce a sequence on tasks and can step the developer through those tasks by waiting for each task to complete before proceeding.


Cheat Sheet Selection

Select the cheat sheet to open:

[✓ Select a cheat sheet from the list:]

- Java Development
  - Create a Hello World application
  - Create a Hello World SWT application
- Plug-in Development
  - Creating a Rich Client Application
  - Creating an Eclipse Plug-in
- Team/CVS
  - Check out a CVS project
  - Merge CVS branches
- topPROCESS TQP

Follow the steps for merging changes from one CVS branch into another.

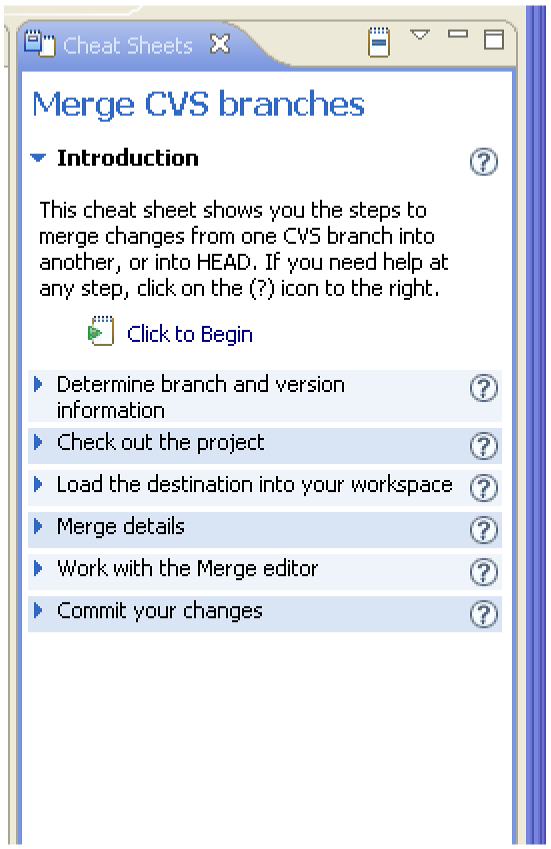
Select a cheat sheet from a file:

Browse...

Enter the URL of a cheat sheet:

OK Cancel

Figure 2 - Cheat Sheet menu


Cheat Sheets

Merge CVS branches

Introduction

This cheat sheet shows you the steps to merge changes from one CVS branch into another, or into HEAD. If you need help at any step, click on the (?) icon to the right.

Click to Begin

Determine branch and version information

Check out the project

Load the destination into your workspace

Merge details

Work with the Merge editor

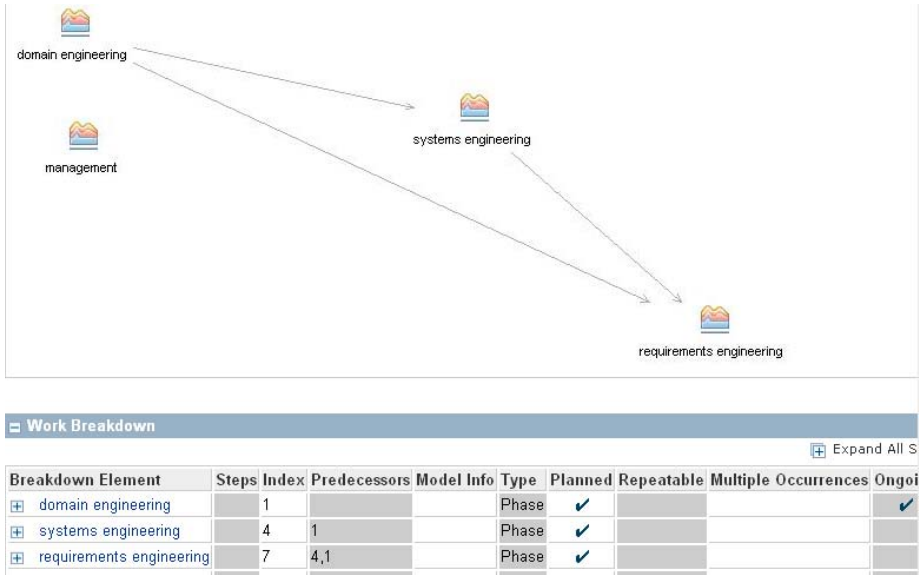
Commit your changes

Figure 3 - A Cheat Sheet

EPF – The Eclipse Process Framework (EPF) provides a facility for describing software development methods [Eclipse 09b]. EPF is an instantiation of the Software Process Engineering Meta-model (SPEM) [OMG 08]. These constructs are captured in a set of templates that are instantiated and linked to define a software development method, not just a process. A method is a comprehensive set of practices that include processes, models, and tools needed to support the development of a software-intensive product.

EPF supports parallel definition of methods by a distributed workforce. Content is defined within a Method Plug-in, a module for encapsulating information with a well-defined interface. Given a base set of common definitions delivered in a Method Plug-in, other Method Plug-ins would be created by each core asset developer to contain their attached process. These new Method Plug-ins “extend” the base Plug-in. This provides a set of attached processes for the product line coordinated around the common definitions and common understanding.

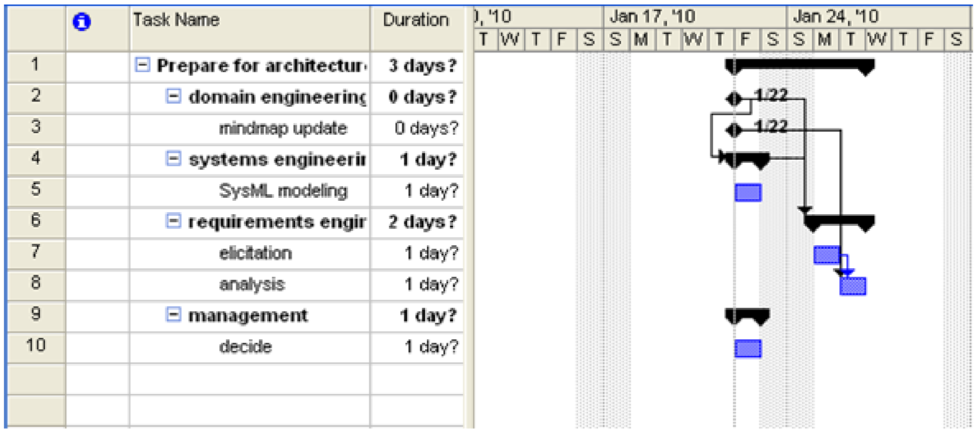
The attached process for an asset can be provided to the product builder in several ways. The complete method description can be provided as a website of hyper-linked information, Figure 4. EPF supports a rich text editor that allows for formatted text, URLs, and images in the text. It also supports diagrams such as UML activity diagrams.



Breakdown Element	Steps	Index	Predecessors	Model Info	Type	Planned	Repeatable	Multiple Occurrences	Ongoi
domain engineering		1			Phase	✔			✔
systems engineering		4	1		Phase	✔			
requirements engineering		7	4,1		Phase	✔			

Figure 4 - EPF published website

A detailed process description can be exported to tools such as Microsoft Project, Figure 5. The information from EPF is transformed into formats native to Project.



		Task Name	Duration
1		Prepare for architectur	3 days?
2		domain engineering	0 days?
3		mindmap update	0 days?
4		systems engineerii	1 day?
5		SysML modeling	1 day?
6		requirements engir	2 days?
7		elicitation	1 day?
8		analysis	1 day?
9		management	1 day?
10		decide	1 day?

Figure 5 - Schedule exported from EPF into MS Project

Embedded Documentation –It is always difficult to motivate developers to create documentation, but documentation that is an integral part of the asset is somewhat easier to motivate. For the code, techniques such as javadoc annotations have been around along time. In document preparation tools, such as word processors, there are often techniques such as hidden text or templates that can be used in an unobtrusive way to provide information. The annotations are visible in one mode of the product but invisible in other modes. This approach simplifies version control and trace back so that documentation is easy to keep in synch with the content.

The tools used to create and perform the attached process need to be appropriate to the context and content of the production method. Some product line organizations use a largely traditional manual method for building products and the attached processes will be text descriptions. For a more automated production method the attached processes will tend to be scripts that execute the automated workflows.

5 EFFECTIVE PROCESSES

In discussions about creating attached processes the inevitable argument is “I don’t want to spend time creating things that no one will use.” To which I respond, “if they are beneficial people will use them and if they are not, change them so that they are.” In a previous column I stressed that this is not “unnecessary work” because the processes pay for themselves [McGregor 08]. The attached process reduces the amount of time the product builder must spend preparing to use the asset.

There is no end of advice on how to make processes effective in general, but specifically what does it take for attached processes to be effective?

Fits seamlessly with the production plan – The production plan has variation points (yes, even the plans have variation points) that correspond to the choices to be made among core assets. Each variation point will define the requirements for the variants, in this case the attached process that is to be integrated at that point.

Provides actionable instructions – An attached process needs to be a step-by-step, complete set of precise instructions. I have seen processes written as “guidelines” or as a “conceptual” view of how to use the asset. These may have their place but it is not in the attached process.

Provides complete information for use of the asset – By complete information I mean sufficient information for the product builder to resolve all variation points of the asset.

Communicates effectively with the product builder – The typical advice “Write for the Reader” applies here, even if the process is automated. The attached process states exactly what the product builder is to do. If the product builder is an end-user, assembling a group of plug-ins, the process should be understandable to the product builder. If the user of the attached process is an automated process, the attached process is a script that communicates effectively with the tool it is intended to direct.

Facilitates understanding of the asset more easily than personal examination of the asset – The attached process should provide just what is required to use the asset, not to maintain the asset. If the process provides too much information it take longer to understand the asset using the attached process, that is not effective.

Lastly, an attached process should be subject to review as a part of the core asset package. Having many eyes on the information is the best means to ensure that the deployed process really is effective.

6 SUMMARY

Attached processes make a valuable contribution to the core asset base of a software product line organization. The attached process looks to the future. It is a means for the core asset developer to communicate with the product builders who will use the asset in the future. In many software product line organizations some of the product builders are either not co-located with the core asset developers or even outside the company that owns the core assets. The attached process enables the product builder to use the core asset much more quickly. The attached process is a strategically significant element in achieving the order-of-magnitude gains in productivity that the software product line strategy can deliver.

REFERENCES

[Chastek 02] Gary Chastek and John D. McGregor. Guidelines for Developing a Product Line Production Plan, CMU/SEI-2002-TR-006, 2002.

[Eclipse 09] Eclipse Foundation, www.eclipse.org.

[Eclipse 09b] Eclipse Foundation, Eclipse Process Framework, http://www.eclipse.org/epf, 2009.

[McGregor 05] John D. McGregor. Context, http://www.jot.fm/issues/issue_2005_09/column4/, 2005.

[McGregor 06] John D. McGregor. Planning before plans http://www.jot.fm/issues/issue_2006_03/column3/, 2006.

[McGregor 08] John D. McGregor. Agile Software Product Lines, Deconstructed, http://www.jot.fm/issues/issue_2008_11/column1/index.html.

[Microsoft 09] Welcome to the Windows Ecosystem Readiness Program, http://www.microsoft.com/whdc/Win7/default.mspx, 2009.

[OMG 08] Object Management Group, Software & Systems Process Engineering Metamodel specification (SPEM), Version 2.0, http://www.omg.org/spec/SPEM/2.0/ 2008.

[Topcased 09] Topcased, www.topcased.org.

About the author

Dr. John D. McGregor is an associate professor of computer science at Clemson University, a visiting scientist at the Software Engineering Institute, and a partner in Luminary Software, a software engineering consulting firm. His research interests include software product lines and component-base software engineering. His latest book is *A Practical Guide to Testing Object-Oriented Software* (Addison-Wesley 2001). Contact him at johnmc@lumsoft.com.