JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 6, July-August 2008

Mix and Match

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

Abstract

Recently there has been much interest in blending successful development methods such as agile development and product line engineering into an even more successful hybrid. I will consider one technique for accomplishing this blending. I will briefly present an example that will be discussed in more detail at the Software Product Line Conference (SPLC) in September. In this issue of Strategic Software Engineering I want to discuss how to achieve synergy between methods and how to benefit from the results.

1 INTRODUCTION

In the 1800s, Mendel experimented with cross-breeding plants as a means of developing a new plant with the desirable characteristics of both. Many experiments were tried and many failed, partly because Mendel was also trying to understand the mechanisms at work. Some of the resulting combinations were worse than the original plants, others carried the best traits of both. Mendel was able to discern some general rules about the relationships of traits of parents and the resulting traits of offspring. Genetic engineering was born.

Over many centuries humans have struggled to co-exist despite their differences. People of different religions, different skin colors, and different sexual orientations have sometimes been successful at integrating into a unified community but in some cases have partitioned their communities so that there is a separate place for each group rather than mix together. Diplomacy was born.

Now there is interest in what happens if development methods based on different ideologies are combined. Most recently there has been interest in "agile product line engineering." Boehm has placed agile development and product line engineering at opposite ends of a continuum[Boehm 02]. Combining two development methods has elements of genetic engineering and diplomacy and some other disciplines as well. Development involves technologies which behave predictably and developers who don't. There are many questions about different means of combining and the results of those combinations. Method engineering may have some of the answers.

In all of these situations the intent is to develop something new and better from the combination. In the case of development methods, most often the resulting combination will have predictable traits but occassionally there will be an unexpected result. This unexpected behavior emerges from the combination and is seldom

Cite this article as follows: John D. McGregor: "Mix and Match", in *Journal of Object Technology*, vol. 7, no. 6, August - September 2008, pp. 7-13 http://www.jot.fm/issues/issue_2008_05/column2/

predictable since it is not part of the specified behavior of either method. The emergent behavior is a bi-product of the interaction of the activities that have been brought together. For example based on componant integration, the emergent behavior may be a never terminating loop caused by one thread of each of the two activities interacting in an unanticipated way. Sometimes only a detailed analysis will identify the interaction.

In this issue of Strategic Software Engineering I am going to discuss some options for combining development methods and the implications of these approaches. I will also illustrate with examples from several case studies. Ultimately I will show the strategic influence of these combinations on the organization. Throughout I will use the example of merging agile methods and product line engineering as the driving motivation.

2 THE STARTING POINT

One of the problems with discussing combining methods is coming up with a precise description of the methods being combined. Extreme programming, Scrum and many other names are applied to various flavors of agile development. They share a common set of values that were codified as the Agile Manifesto [Agile 08]. These values, which will serve as my definition of agile, include:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

The manifesto elaborates these values into 12 principles, which I will not list here. These principles provide slightly more specific direction, but the values and the principles are ones with which few software professionals would argue. It is often a matter of degree. Everyone would agree that individuals are more important than processes, but the arguement can be made that processes and tools can save sufficient time to focus on individuals. The result is that many organizations claim to be following agile practices but without achieving the essence of the method.

The software product line approach of the SEI and the product family and product population approaches used in Europe share much but vary from each other. Even within the SEI's approach, different organizations will emphasize different ones of the twenty-nine practice areas resulting in very different total approaches. I will use the twenty-nine practice areas in the SEI's Framework for Product Line Practice as the definition for a software product line.

All of this variety is healthy but makes our job more complex. Each variation is someone's attempt to devise a method that more exactly addresses their development problems. The breadth of definition means that we need to clearly identify the attributes we are assuming in our discussion. For example, for the purposes of this column using agile and the four manifesto values may suffice but for a development project, which is defining its own method, a much more specific definition is needed to define an operational method.

3 SOFTWARE PROCESS ENGINEERING META-MODEL

The Software Process Engineering Meta-model (SPEM) provides a means by which method definitions can be represented and systematically analyzed and merged[SPEM 08]. The meta-model provides the basis for defining process modeling languages which can then be instantiated for specific situations. The meta-model is structured to support iterative, incremental process modeling. In particular, SPEM defines three types of associations between definitions: Extends, Replaces, and Contributes. These associations are used to derive a new definition from an existing one. The default association is Inherits, in which behavior defined in a definition is carried over into the new definition without modification. SPEM also defines the concept of a method plug-in that provides ways to integrate existing definitions into a single framework.

The Eclipse Process Framework Composer provides tool support for manipulating methods that are modeled using languages based on SPEM[EPF 08]. The tool supports the development of plug-ins, similar to Eclipse plug-ins, that can be combined in many ways to produce methods specific to the project at hand. The tool also supports the publication of these process models as a set of web pages that are hyper-linked.

4 METHOD CROSSBREEDING

So now lets think about crossbreeding two existing methods. First we will do this without regard to what the two methods are and then we will consider specific examples. We will refer to method A and method B as the two existing methods we wish to crossbreed and method C as the method that results. Although it often does not matter, we will assume that method A will form the basis for method C with elements of method B integrated where appropriate.

Method engineering defines several ways in which development methods can be combined[Cossentino 04]. In fact combining two methods usually will require multiple approaches at various points in the definitions. Basically the possible approaches roughly parallel the three associations in SPEM mentioned in section 3: extends, replaces, and contributes.

Extends

An activity in method A is enhanced with actions from a similar activity in method B. For example, an agile feature modeling activity might be extended to explore constraints among the features as required by the product line development method. The resulting method C has the traits of method A and the traits of method B related to the constraint definitions, but attributes such as the length of time to complete feature modeling has changed.

Replaces

An activity in method A is replaced by a similar activity from method B. For example, an agile method for feature modeling can be used in place of the currently used feature modeling technique within a product line analysis method provided:

- the replacement method does not require changes to the processes that supply information to the feature modeling process,
- does not interact with processes that operate concurrently with the feature modeling, and
- does not require any changes to the processes fed by the output of the feature modeling method.

In a way this is like keeping the specification of a component constant but changing the implementation. Method A retains its basic traits but the feature modeling activity may take a different amount of time or it may be less accurate so there will be an impact on the traits of method C. The method engineer should determine whether that impact is positive or not.

Contributes

An activity in method A is enhanced by adding some of the steps from a similar activity in Method B. The resulting method C has all the traits of A and the added traits from those steps. The new method may take longer to apply than A. The new method should produce at least the same level of detail and might produce more output than A.

Summary

Extends and Contributes alters the method by additions while Replaces removes part of one method and replaces it with material from the other method, but these are just the means to the end. The goal we are after is a better development method than the original two that were combined. This is judged by the business goals that are enhanced by the hybrid method. For example, a product line optimizes the delivery of a set of products based on a set of reusable assets. Our agile product line should be able to optimize delivery of those products but be more flexible about exactly what those products are until later in the development cycle. As always it is a matter of the method engineer making trade-offs. In this case, code will be reworked later in the cycle, thereby increasing waste, but the increase in flexibility will allow us to more completely satisfy customers. I have used common sense reasoning to intuit about a few of the resulting properties of the new method. Much research is needed to determine precisely how the associations affect the new method.

5 AN EXAMPLE

We can now return to our original query. How might we blend an agile method with a software product line method to produce an agile product line method. I will use the SEI's Framework as the definition of product line and the 4 manifesto values as the definition of agile. Since I view the software product line method as more encompassing, I will structure the discussion using the three categories of product line practice areas, organizational management, technical management, and software engineering, as the base method and add the agile method into it. I will focus on the overlaps between the methods in each practice area.

The organizational management practice areas for a software product line do not have major interactions with the agile method, but there is clear evidence that these practices are necessary for product line success [Carbon 06][Noor 08]. The one specific practice I want to discuss is Customer Interface Management (CIM). Unlike traditional development in which developers are often told not to talk directly to customers, agile software development requires a very close relationship between developers and customers. In fact one of the principles of agility is that developers must work with "business people," who often will be customers, daily. The product line CIM practice calls for clear communication of the scope – limits of functionality in the products - of the product line to customers. An agile product line would have to modify the customer orientation of each method. The CIM activities of the product line will be ***extended*** to include a closer collaboration with customers but within the overall constraint that any requirements outside those of the product line will take longer and cost more.

The technical management practice areas have more overlap with the typical agile method definition. I will consider three specific practices:

- Technical planning in a product line typically takes a longer term view than a typical agile plan. The technical planning practice would probably be ***replaced*** with a modified process that uses a two-tiered approach. The longer term view would address the scope of the product line and how to coordinate the production of products from the core assets. The shorter term view would be that of a specific product development team.
- The Process Discipline practice area should be ***inherited*** directly as defined in the product line method. The process discipline calls for definition of the appropriate processes. Therefore, the organization operating the hybrid product line would use a standard approach to process definition and the personnel assigned responsibility for the process will follow it.
- The Tool Support practice area has a significant counterpart in the agile method. Agile projects use modeling tools to generate tests, source code, and documentation. Many actions in a product line method are automated as well. In this area the agile method would ***contribute*** tools and their practices.

The software engineering practice areas have two levels of application in a product line method. The core assets, which must address the entire product line, represent a large number of software engineering practice areas. These assets are delivered to product development teams where they instantiate each asset using the choices of variants appropriate for their assigned product. I will not try to address all of the software engineering practice areas. I will briefly discuss several:

- Requirements Engineering – This is a practice area where the customer-centered approach taken by agile organizations is particularly important to support. Typically this will simply be an ***extension*** of the product line requirements engineering method. The feature-driven approach taken by many product line organizations should also be part of the hybrid method because it captures the initial view of the results of the commonality and variability analysis.

- Architecture Definition – The product line architecture is an essential asset for guiding many of the activities of the organization. In an agile project the architecture is typically not as detailed as the product line architecture. The new method would have an architecture definition practice that ***replaces*** the product line and agile practices. The new practice would focus on essentials of structure and commonality/variability but not in as much detail as the product line practice.
- Testing – This is a practice area where agile development and product line development are already in harmony. The test-first method for component unit testing is used widely in both agile projects and software product lines. Both agile and product line organizations use extensive tool support for testing at most levels. In this case the merged method would simply ***inherit*** the testing practice definition of one of the existing methods.
- Software System Integration – The important issue in this practice area is providing working software continuously. Agile projects deliver working versions of products, albeit with very limited functionality at first, to customers from the earliest stages of the project. Product line organizations do not produce products in the earliest stages. They spend some time producing core assets first and then integrate. Once a set of core assets is available, then products can be produced rapidly. The agile integration approach would be ***contributed*** into the product line production process used by product teams but not in the core asset development process.

6 SUMMARY

The ability to blend existing methods, which exhibit specific characteristics, into a new method that exhibits some set of those traits, is a strategic advantage. It allows the organization to tailor how different customers are managed, to control product production, and to adjust levels of certain quality attributes.

The SPEM provides us with a useful tool for engineering methods. To me that means I can use first order principles to reason about various attributes of a process. This makes combining multiple development approaches a deterministic operation. I can know what traits I want the resulting method to have and I can combine the methods knowing I will achieve what I am after.

Combining agile and product line engineering is a potentially very powerful union. Although much research remains to be done, early experience is showing the benefit of this combination. This discussion will be continued in a special session at the Software Product Line Conference (SPLC) in Limerick in September 2008 [SPLC 08]. The working session will engage attendees as active participants. Join us.

7 ACKNOWLEDGEMENTS

Thanks to Dr. John Hunt of Covenant College for his valuable comments on this issue.

REFERENCES

[Agile 08] Agile Manifesto. www.agilemanefesto.org, 2008.

[Boehm 02] Barry Boehm. Get ready for agile methods, with care. IEEE Computer, 35 (1), 64 – 69.

[Carbon 06] Ralf Carbon, Mikael Lindvall, Dirk Muthig, Patricia Costa. Integrating Product Line Engineering and Agile Methods: Flexible Design Up-front vs. Incremental Design, First International Workshop on Agile Product Line Engineering, 2006.

[Cossentino 04] Massimo Cossentino and Valeria Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering, Composition of a New Process to Meet Agile Needs Using Method Engineering, 2004.

[EPF 08] Eclipse. www.eclipse.org/epf.

[Hanssen 08] Geir K. Hanssen and Tor E. Fægri. Process fusion: An industrial case study on agile software product line engineering, Journal of Systems and Software, 81 (2008), pp. 843 – 854.

[Noor 08] Muhammad A. Noor, Rick Rabiser, and Paul Grunbacher. Agile product line planning: A collaborative approach and a case study, The Journal of Systems and Software, 81 (2008), pp. 868 – 882.

[OMG 08] Object Management Group. The Software Process Engineering Meta-model, 2008.

[SPEM 08] Software Process Engineering Meta-model, www.omg.org, 2008.

[SPLC 08] Software Product Line Conference (SPLC), www.splc.net.

About the author

Dr. John D. McGregor is an associate professor of computer science at Clemson University 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.