JOURNAL OF OBJECT TECHNOLOGY

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

Vol. 7, No. 8, November-December 2008

Agile Software Product Lines, Deconstructed

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

Abstract

There was much interest at this year's Software Product Line Conference in how to combine agile and product line techniques. Agile teams seek to address change one product at a time while product line organizations take an investment view by addressing change among a set of products. On the surface there are some seeming contradictions between the methods, but they may not be as different as they are sometimes portrayed. In this issue of Strategic Software Engineering I want to deconstruct product line and agile practices, compare the pieces, and make some suggestions about how to re-construct a hybrid method. I will do this in part by treating agility as a quality attribute of processes.

1 INTRODUCTION

Recently my wife and I went to a restaurant for dinner. On the dessert menu, among other goodies, was a "deconstructed" chocolate cake. The chef had four separate elements that were, to her, the essence of the chocolate cake. Each element was more elaborately prepared than it would have been in the cake but provided the flavors and textures that one would experience. A chef does this to let the diner focus on the "essence" of each element. The diner can choose to eat the elements individually or combine a little of several of the elements in one bite if they wish. I want to do the same with the agile and product line methods with the intention of investigating new combinations of practices that provide new benefits.

Many of the discussions about combining these two have taken both agile and software product lines as monolithic methods and attempted to glue them together. My purpose in deconstructing is to see if there are pieces of each that might work well together and whether other pieces might be dropped while still retaining the essence of each method. To do this I will consider the specified characteristics of each but I will also dig into a few implicit assumptions that lurk behind each approach.

There was much interest at the Software Product Line Conference (SPLC) 2008 in the relationships between agile and product line methods. Agility played a role in the practices of Philips HealthCare as described in Luc Koch's keynote address [Koch 08]. A

Cite this column as follows: John Mc Gregor: "Agile Software Product Lines, Deconstructed", in *Journal of Object Technology*, vol. 7, no. 8, November - December, pp. 7-19 http://www.jot.fm/issues/issue_2008_11/column1/

large group participated in a working session, which I chaired, that brainstormed ideas regarding agile and product lines. In that session we raised many questions and started several conversations that I hope will carry on to next year's conference. I will rely on these and other sources as I consider how these techniques could be blended.

Let me start right away by making explicit a few of my personal assumptions about this topic. First, I could not care less about creating an agile software product line method if it does not improve how we are developing software intensive products. The cachet of a particular name is irrelevant. Note that when I say "improve" that can mean many things. I am willing to create more waste and scrap if my goal is quicker time to market and creating the waste gets me there.

My second assumption is one that I believe we all hold in common: "No one wants to do extra work," but what "extra" means is part of what separates us. I believe it is a matter of perspective and context. A product line perspective encompasses a set of products built over a period of time while the agile perspective is more focused on single product development. Doing work on a component that is ***scheduled*** to be used in a product even though we have not started assembling yet is reasonable and essential to both the agile and the product line staff. Working on a component that is ***anticipated*** to be used in a product is essential work to a product line person but extra work to the agile person. A product line organization operates in a context of sufficient stability that planning the development of several products is not an exercise in futility.

My third and, so far, final assumption is that the strategic levels of reuse is what provides the productivity and time improvements that make the software product line approach useful. Any new method that reduces the amount of existing assets used in products will reduce the benefits of developing a product line.

In a previous column, **Mix and Match**, I explored techniques for combining processes, models, and tools into a coherent development method [McGregor 08]. I used agile and product line practices as an example as I described the mechanics of method engineering. In this column I will dig deeper into the conceptual content of each method and focus on the issues related to the development activities rather than the issues regarding combining practices.

Neither "agile" nor "product line" refers to a single universally accepted definition. Rather each refers to a class of methods that share certain characteristics. I will briefly characterize each class of methods trying not to alienate too many people along the way. Then I will describe some combinations of pieces that might prove strategically significant.

2 AGILE MANIFESTO

As the name implies, "agile" techniques exist in a development environment that is tuned to respond rapidly to changes in product requirements. The agile community is a diverse group but some of them have collaborated on an "agile manifesto" [Agile 08]. This manifesto consists of twelve principles listed in Table 1, in accordance with the copyright

notice the entire description is listed in the table. These principles seem quite straightforward and self-explanatory to me and serve as a useful deconstruction of the agile approach.

Table 1 Agile Principles

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Business people and developers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Working software is the primary measure of progress.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Continuous attention to technical excellence and good design enhances agility.
Simplicity--the art of maximizing the amount of work not done--is essential.
The best architectures, requirements, and designs emerge from self-organizing teams.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

From a software engineering perspective I can't disagree with any of those statements. I would say that for me "best" is a relative term driven by the business goals. From my personal product line perspective, I am willing to tradeoff absolute adherence to any one of these principles in order to achieve an important business goal. For example, face-to-face communication may be best for clarity but there are times when budgets stretched by rising airfares constrain us to either an internet meeting/teleconference or no meeting at all. Insisting on working daily with a business person may result in an inexperienced, read that as cheap, person being assigned. I would prefer the experienced person occasionally to the newbie continuously.

In the title of this column, "Agile Software Product Line," agile is used as an adjective. It is an attribute of the product line. A quality attribute, sometimes referred to as a non-functional requirement, is difficult to characterize as concisely as a functional requirement. Quality attribute scenarios provide a means of describing an attribute by means of multiple examples. In Table 2 I show a quality attribute scenario for agility using the style from [Bass 98]. I have given this one a product line flavor by talking about carryover from existing to new features. Time did not allow me to develop a large set of scenarios, which is what would be needed to fully define agility, but hopefully this one will trigger thoughts of other scenarios.

Table 2 An Agility Scenario

Agility: A user, after working with the product for a while, suggests a new feature that is similar to one of the features she used in the product.	
Stimulus	A new feature request
Source of stimulus	A user of the prototype
Environment	A new product sandbox
Artifact	The prototype
Response	The new feature is added
Response measure	The feature is added in time for the next release, faster than if the feature were unrelated to any other features in the previous or current products.

3 PRODUCT LINE APPROACHES

The early literature contained two different terms: “product family” and “product line.” There was even a series of workshops on product families and a series of conferences on product lines. Recently the workshop series and the conference series merged into SPLC, the Software Product Line Conference. This merger was in recognition that the term “software product line” subsumed product families.

In short, the concept of a product family recognizes the commonalities and variabilities among a set of potential products. A software product line places those commonalities and variabilities into an organizational context and considers what actions and resources are needed to actually be able to produce those similar products.

Strategic reuse is a key concept. The software that implements the common features is obviously shared by all. The variations will usually be shared by several products. This is the blessing and curse of reuse. Variability management and the management of assets across multiple products is much of what product line research is about. Avoiding “clone and own” is critical for the long term health of the assets and yet it is difficult to do well.

There are three widely recognized approaches that a software product line organization can pursue with respect to investment in core assets. No one of them is the best approach in all cases, it depends on the context in which the product line organization operates. The three are:

- Proactive – In the proactive approach the reusable assets are developed prior to any product development. A product line requirements model and an architecture are created and from these assets code assets are derived. This approach is efficient in a very stable domain where the organization has lots of experience. The biggest risk is that over time, before some products in the product line can be built, changes in the business climate or in the domain and product definitions will render useless some of the assets already created.
- Reactive – In the reactive approach the reusable assets are harvested from products after they are built and deployed. Initially the product is built like any single-product development effort. As other products are built they use assets harvested from the products that have been built so far. The set of reusable assets evolves into a more useful collection over time. This approach reduces the risk that assets will become obsolete. Every asset is used at least once. The risk is that some short fuse business opportunities will be missed because product production is not as fast as it could be. There is also the risk that lack of a product line architecture will result in lots of reworking of assets to make them suitable for future products. Note that this is not the reactive approach described by some agile methodists in which they “react” to each product as a new, and relatively unique, undertaking.
- Incremental – The incremental approach is a compromise between the two extremes. The set of assets is built in scheduled increments. The increments are

usually defined to provide assets needed for a set of products scheduled to be produced in the near future. The risks of the other two approaches are present here in reduced form but still present.

Finally, let me give a bit more detail about the product line approach. The Software Engineering Institute (SEI) has developed a Framework for Product Line Practice [SEI 08]. This framework describes 29 practices, shown in Table 3, that affect the success of a software product line organization. This would be a very good deconstruction of product line practice but it is a few more pieces than I want to work with. Rather, I will use two different decompositions of the practices areas as my basis for discussion.

One way of slicing the method is to group the practice areas into three categories:

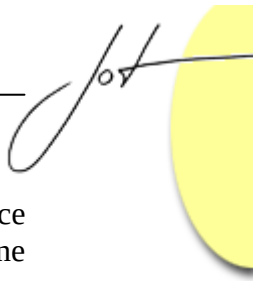
Software engineering – The practices in this category relate to technical issues about identifying variability and building products.

Technical management – The practices in this category relate to managing a development environment in which multiple products may be in progress at the same time and variants within a single variation point may contradict each other.

Organizational management – The practices in this category relate to managing a diverse organization with a single set of goals but with a variety of perspectives on those goals.

Table 3 Practice Areas

Software engineering	Technical management	Organizational management
Architecture Definition Architecture Evaluation Component Development Using Externally Available Software Mining Existing Assets Requirements Engineering Software System Integration Testing Understanding Relevant Domains	Configuration Management Measurement and Tracking Make/Buy/Mine/ Commission Analysis Process Discipline Scoping Technical Planning Technical Risk Management Tool Support	Building a Business Case Customer Interface Management Developing an Acquisition Strategy Funding Launching and Institutionalizing Market Analysis Operations Organizational Planning Organizational Risk Management Structuring the Organization Technology Forecasting Training



Another possible deconstruction of the product line approach is three pieces that slice across what we have already sliced. Think for a few minutes about product line development as asset building, product building, and management.

Asset building – This role is the "long" view. What are the pieces that can be reused?

Product building – This role is the short-term, get-it-out-the-door view. What is the least work I can do and get this product to work?

Management – This is the investment view. How can we track the resources invested in the assets and determine their ROI?

Table 4 provides a view of the dice that results from all this slicing.

Table 4 Deconstruction of the product line approach

	Software Engineering	Technical Management	Organizational Management
Asset building	Technical asset building such as code modules	Creating templates for plans, reports, and other technical assets.	Developing processes for identifying and initiating product lines
Product building	Assembling modules and writing code necessary for the unqiue behavior of a product	Scheduling of products to maximize market impact and effectively use staff	Establishing product roadmaps that utilize the time-to-market advantage of the product line approach
Management	Continual evaluation of the return on investment (ROI) of the technical activities, particularly whether the assets are useful to the product builders	Evaluating the ROI for the management assets, particularly whether the assets provide technical managers with the tools to manage engineering activities	Determines whether asset and product building organizations are effective in their roles

4 AGILE PRODUCT LINE METHODS

I will assume that the reader is familiar with method engineering and, if you are not, I suggest reading Mix and Match where I talked some about using the Eclipse Process Framework to compose methods and to [McGregor 04] where I discussed goal-driven

method engineering. In this section I will focus on characteristics, common goals, and common tasks that will serve as points at which method fragments could be blended.

Dimensions

I disagree with those who define a dimension from "agile" to "plan-driven" and place product line methods at the plan-driven end of the spectrum. I disagree because I do not think that agile is the absence of plans anymore than I think a product line organization would drive off a cliff because a plan said to. Besides, to determine the correct method to use it is important to first understand the business context of the products not the content of the methods.

Three dimensions are of interest in characterizing the business environment with regard to the suitability of an agile product line approach. The degree of commonality among the products determines the percentage of the product that could come from reuse. The higher the commonality, the more potential benefit there is from an asset-based approach and the less benefit from custom building each product. Depending upon the method chosen, the organization may reach its full potential for reuse faster than with other methods. Some reuse is commodity reuse, graphical user interface controls for example which can be realized very quickly. For this dimension I am more interested in feature reuse. A reactive approach will reach full reuse potential much more slowly than a proactive approach, but before you jump to conclusions, you never get something without giving up something.

The second dimension is the volatility of the relevant domains. The more rapidly a domain changes the less value there is in an asset-based approach is and the more value there is in a custom built approach. The multiple domains relevant to a product line have differing levels of volatility. Commodity domains such as user interfaces have little volatility. The volatility of domains relevant to the content of the products is the main issue.

The third dimension is magnitude. The size of products, teams, and the organization are all factors. Agile techniques have proven successful on small projects but have often encountered problems on large projects. This is often considered a negative for application to product lines. However, the products in a product line may be small and the development of each well within the reach of an agile method. There have been numerous efforts to scale up agile methods, but at this point, this is a significant characteristic in building a method.

Locating the points along these dimensions where an organization is located helps determine the method that should be fabricated. For example, building a set of products related to "green" technologies will require a more agile scoping activity than one for mp3 players. New product ideas will come rapidly early in the life of a domain such as "being green" and some of them will be sufficiently unique to be disruptive and invalidate the assets built previously.

Characteristics

Lets explore some ideas that are useful in defining a method.

Both agile and product line methods are collaborative. Agile methods are based on collaboration between product users and the developers. Product line methods expect collaboration between the asset builders and product builders. These collaborations serve the purpose of a feedback circuit. It provides the opportunity for continual, hopefully minimal, redirection.

Agile methods accept changing requirements, in fact they are formulated to encourage change, and handle it when it happens by renovating code. Product line methods accept variable requirements by anticipating them and planning for them by including variation points in the design of each asset. Not only does this handle anticipated change, it positions the product line to accept unanticipated changes, better than traditional approaches, since some of those changes will coincide with variation points. It is easier to make changes by adding new variants than to introduce changes in code in which no provision for change has been made. It is the amount of volatility, unanticipated change, that can keep a product line organization from achieving its goals.

Both agile and product line methods operate within a scope. Scoping is an explicit activity of a product line organization. The scope describes the products that are part of the product line. An agile project works within an implicit scope defined by the domains represented by the customers that are the source of requirements. When recognized, the limitation imposed by the scope constrains the changes that are likely and in turn this can be used to focus the reaction to those changes.

Both agile and product line methods "*maximize the amount of work not done.*" Agile methods do it by postponing work until it is needed while product line people do it by systematically anticipating what will be needed and then creating assets for specific purposes within the specific scope. When a test plan is made into a core asset by adding variation points, it is because of the products within the scope and in anticipation of saving work on other test plans later.

Agile methods produce working software early in the development of a product. Product building teams in a product line do so as well, by assembling and configuring core assets. In fact the reason for producing working software early is so that users can get direct experience with a product and give feedback sooner than later. Product line organizations often have complete example products very early in the life of the product line, before some specific products are even started.

Engineering a method

Based on what I have said so far it should be clear that no single method can be defined that is "the" agile product line method; rather, we can describe some method fragments that could be used in the correct situations. Both methods will have to be tailored to support the integration. Several industrial examples of tailoring agile practices have been published. For example Motorola found it useful to create a baseline architecture to guide

their tailored version of XP. This may be a less detailed architecture than typical for a product line, but it could provide sufficient guidance to support the creation of core assets [Lindvall 04]. Nokia found that minimizing the interface between agile and non-agile teams improved communication [Lindvall 04]. Trinidad et al describe a tailoring of feature modeling to make it more agile [Trinidad 08].

I want to make an initial proposal for defining an agile software product line method by starting with the software product line method and adding the quality of agility where appropriate and possible. I propose to start with a skeletal framework. I will start with the SEI's framework. This is a natural starting point since the framework covers both the business and technical perspectives; however, an organization already using agile methods would probably find it more natural to start from their base and add product line qualities and concepts.

The SEI's framework specifies practice areas and then defines specific practices within each practice area. Within that framework several approaches are possible. I will consider two main approaches.

Micro approach

First we could examine each of the 29 practice areas and apply tactics to make some of the specific practices more agile. Agility is enhanced in a number of places although there may be no identifiable process that is agile. In Mix and Match I addressed several specific practices that are compatible with an agile approach. These could be the starting point for making several specific agile practices.

I will take one practice area, testing, as an example. Some time ago we published a technique we named the Parallel Architecture for Component Testing [McGregor 96]. In this technique the test assets are built using the same architecture as the component under test. Our research has shown that this is a cost-effective technique that supports evolution of the asset. The common architecture provides a traceability that allows the developer to more rapidly refactor and revise the asset and the associated tests.

Macro approach

A second approach would learn from the Nokia experience and identify the place where a narrow interface is possible and allow tasks on one side of the interface to use an agile approach. In a product line organization, the most narrow interface is that between the core asset team and the product building teams. It is narrow in the sense that many product lines deliver the core assets as binary components that can only be used as implemented. Many product line organizations also deliver the core asset base as a single package. Product line organizations provide a feedback mechanism between the teams that provides for error reporting and new feature requests. Feedback mechanisms are also a standard design for governing the speed with which two processes communicate. Making the feedback mechanism more agile would allow the core asset development process to become more agile.

Following this approach raises the question of whether the core asset team or the product building teams use an agile approach? Each product team has its own interface

with the core asset team and could use an agile process independent of the other product teams. The size of the product might be a determining factor in that case. A factor in favor of the product team using an agile approach is that the product team has the most direct interface with the customer for a specific product. A factor against using an agile approach is that product line organizations in a very stable, well-understood domain should be able to use a highly automated, perhaps even waterfall, approach to product building which does not require the highly-motivated personnel used in an agile project.

The core asset team might use an agile approach since, especially early in the lifetime of the product line, developing a core asset is an exploratory process that could benefit from those highly-motivated people. There has been success with evolving the core assets to full functionality. The interface between the two types of teams might also be a target for being made more agile. I know of several core asset teams that provide a help desk for product developers to call for help in using the core assets. In many cases though any defects reported can still take a long time to repair and the help desk is viewed mainly as a flow of information out to product teams rather than a collaboration. Changing the way core assets are delivered to release fixes as soon as each is ready could speed up product development, but broadens the interface between the teams making it more difficult to manage effectively.

Frameworks other than the SEI's could be used as the basis for an integrated method as long as there are clear interfaces to each of the variation points in the framework. Qumer and Henderson-Sellers provide a framework for agile development that defines several facets that could be used as a basis for supplementing the product line framework with agile practices [Qumer 08].

5 SUMMARY

There are evident synergies between the agile and software product line methods, but competing philosophies make their integrated use difficult. A number of experiments have been, and are being, conducted with various combinations of these methods [Koch 08] [Wessilius 08]. Although no single method has emerged, the tailored instantiations of these methods are able, in many cases, to retain the advantages of each separate method and enhance the practices of the organization.

The tailoring of each method must ensure that the basic characteristics of each method are retained. Considering the goals of each task and the differing perspectives of each method when blending tasks has allowed organizations to effectively deploy hybrid methods that are agile and yet asset-based at the same time. By engineering these hybrid methods specifically to meet the business goals of the organization, the development teams can make a strategically significant contribution to the organization.

ACKNOWLEDGEMENTS

I want to thank Mark Dalgarno of Software Acumen and Yaser Ghanam of the University of Calgary for their insightful comments and helpful suggestions, especially in such a short time frame.

REFERENCES

[Agile 08] http://agilemanifesto.org/, 2008.

[Bass 98] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice, Addison-Wesley, 1998.

[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.

[Clements 05] Paul Clements, John D. McGregor, and Sholom G. Cohen. The Structured Intuitive Model for Product Line Economics (SIMPLE), Software Engineering Institute, CMU/SEI-2005-TR-003.

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

[Koch 08] Luc Koch. Keynote address at SPLC 2008.

[Lindval 04] Mikael Lindvall, Dirk Muthig, Aldo Dagnino, Christina Wallin, Michael Stupperich, David Kiefer, John May, Tuomo Kähkönen. Agile Software Development in Large Organizations, IEEE Computer, December 2004.

[McGregor 08] John D. McGregor. Mix and Match, Vol. 7, No. 6, July-August 2008. http://www.jot.fm/issues/issue_2008_07/column1/index.html

[McGregor 04] John D. McGregor. Factors in Engineering Strategically Significant Software Development Methods, OOPSLA Workshop on Method Engineering, 2004.

[McGregor 96] John D. McGregor and Anu Kare. "Testing Object-Oriented Components," Proceedings of the 17th International Conference on Testing Computer Software, June 1996.

[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.

[Qumer 08] A. Qumer, B. Henderson-Sellers. A framework to support the evaluation, adoption and improvement of agile methods in practice, The Journal of Systems and Software 81 (2008) 1899–1919.

[SEI 08] Software Engineering Institute, www.sei.cmu.edu/productlines, 2008.

[Tian 06] Kun Tian and Kendra Cooper. Agile and Software Product Line Methods: Are They So Different? In: 1st International Workshop on Agile Product Line Engineering (APLE'06). IEEE Computer Society: Baltimore, Maryland, USA, 2006.

[Tinidad 08] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Corte´s, M. Toro. Automated error analysis for the agilization of feature modeling, The Journal of Systems and Software 81 (2008) 883–896.

[Wessilius 08] Jacco Wessilius. The Bazaar inside the Cathedral: Business Models for Internal Markets, (2008) 60 – 66.

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.