A Report on the XP Workshop on Agile Product Line Engineering

Yaser Ghanam
University of Calgary
yghanam@ucalgary.ca

Frank Maurer
University of Calgary
fmaurer@ucalgary.ca

Pekka Abrahamsson
University of Helsinki
pekka.abrahamsson@vtt.fi

Kendra Cooper
University of Texas at Dallas
kcooper@utdallas.edu

http://groups.google.com/group/aple
DOI: 10.1145/1543405.1543429
http://doi.acm.org/10.1145/1543405.1543429

Abstract

Software Product Line Engineering (SPLE) promises to lower the costs of developing individual applications as they heavily reuse existing artifacts. Besides decreasing costs, software reuse achieves faster development and higher quality. Traditionally, SPLE favors big design upfront and employs traditional, heavy weight processes. On the other hand, agile methods have been proposed to rapidly develop high quality software by focusing on producing working code while reducing upfront analysis and design. Combining both paradigms, although is challenging, can yield significant improvements. In this workshop, we discussed the challenges, the research questions and the tradeoffs that need to be addressed for such an integration to enjoy success.

Keywords: Software Engineering, agile product line engineering.

Overview

The half-day workshop took its course in the afternoon of Wednesday May 27th, 2009 in Sardinia, Italy. Including three organizers, a total of 17 participants took part in the different workshop activities. Participants were mainly individuals from industry in addition to a relatively smaller number of academics. After a brief introductory note of the motivation behind and the goals of this workshop, attendees were asked to introduce themselves, their professions and their expectations of the workshop. Following that was a one-hour panel discussion guided by questions from the facilitator as well as questions raised by the audience. For the group discussions, a number of topics were solicited from the audience; and after a scoring exercise, three topics were selected. This document is a succinct summary of the main issues, topics and questions discussed during the workshop.

Motivation, Goals & Expectations

Motivation. Software Product Lines (SPLs) are sets of similar, yet not identical, systems developed by an organization based on a set of core assets. SPLs have proved to be effective in lowering the cost of software development, reducing time-to-market, and enhancing product quality. Traditionally, SPLE favors big upfront design and employs traditional, heavy weight software engineering processes. A domain architecture is usually required before individual applications are engineered. On the other hand, agile development methods such as Extreme Programming (XP) have been proposed to rapidly develop high quality software by focusing on developing working code while reducing upfront design and process overhead. Also, anecdotal evidence shows that Scrum is keenly adopted by large software companies basing their product development on SPLs. Scrum bases itself on cross-functional, self-organizing teams working in tightly time-boxed development settings. It is interesting to note that although the goals of the two software paradigms have similarities (time-boxed, high quality, complex software), the solutions to realize the goals seem to conflict and little work is available in the literature to integrate them.

Goal and Scope. The goal of this workshop is to bring together people who are using or want to use agile approaches in the development of SPLs. Also, we would like to open the door for discussing the issues that need to be addressed when thinking about software product lines and agility, and get a sense of the current practices and trends in industry to manage families of products – especially in agile contexts.

Audience Expectations. The expectations of the audience ranged over a wide spectrum. Some attendees were there mainly to collect terminology and understand the definition of agile product lines as opposed to traditional product lines. Others were looking for practical experiences and lessons companies' representatives were willing to share. A number of participants expressed interest in investigating the conflicts between practices in agile methods and SPLs and possible ways to reconcile the two approaches. For some, a raft of specific questions were the motivation to attend the workshop such as: how to get a product line started in an agile organization? What is the cost of adopting a SPL practice? Should we separate core asset teams from application engineering teams? Who should pay for the core assets and the platform development? How to balance long-term vision with short-term plans? While some of these expectations and questions were addressed during the various activities of the workshop, some other were either out of scope or could not make it to the list of selected topics for discussion due to time constraints.

Panel Discussion

Five panelists took part in the panel discussion - three from industry and two from academia:

- Kati Vilkki – Nokia-Siemens Network, Finland
- Steve Fraiser – Cisco Systems, U.S.
- Markku Kutvonen – F-Secure Corporation, Finland
- Frank Maurer – University of Calgary, Canada
- Pekka Abrahamsson – University of Helsinki, Finland

While all panelists acknowledged the importance of embracing both agile principles and SPL practices, there was a disagreement on the extent to which we should attempt to integrate the two paradigms. One view was that SPLs and agile software development have different backgrounds, purposes and practices and thus thinking about integrating them is not wise. Agile methods are about operational not organizational matters. One should try to benefit from both paradigms separately by employing subsets of the practices of each – only those practices that fit best within the organizational scale, structure and culture.

Other panelists disagreed reasoning that current SPL practices can benefit from agile methods not just be used beside agile methods. The rationale is that current SPL practices do not scale down, because they demand a lot of investment upfront which small to medium organizations cannot afford. And, moreover, given the fast-paced technological advancement nowadays, even big organizations might not be able to afford a long time of thinking and planning upfront. As panelists from industry reported, when market circumstances change radically, huge losses result from the upfront investment put in no-longer useful or not-yet-used core assets or platforms. This is why the traditional way of creating and managing SPLs should be revisited to address the new market conditions. Namely, domain engineering should no longer be a separate, long, resource-intensive process, but rather a continuous, iterative, fully integrated exercise within the software development cycle.

Also, platform/core-asset teams and application engineering teams should not be totally autonomous and disconnected entities. Very often, application teams complain that they are paying for core assets that are found to be hard to reuse or integrate; and in some cases they do not even get what they asked for which renders the delivered assets useless. Panelists were on agreement that the structure of separate core-asset and application teams is not efficient. This is why a new way of forming core asset teams and ensuring effective communication channels needs to be considered. For example, it might be more effective to select members of different application development teams to form a core asset team so that the needs of the different products are all addressed fairly. This core asset team should be flexible enough to change its members regularly according to which products are currently in need of which core assets.

The issue of documentation was also discussed. There was a consensus amongst panelists that people do not like to produce documentation and more importantly do not read this documentation later on for a variety of reasons (too long, don't know where to look, easier to ask somebody, not up-to-date … etc). Therefore, minimal investment should go into producing documents; and it would even be better if we can find an alternative such as using test artifacts to represent commonality and variability in the product line. A participant from the audience confirmed this possibility by an experience from their company where they used system tests to cover variation in requirements. Although this was possible and convenient, he asserted, tests were growing too complex very quickly.

Other issues were briefly touched on such as: how the role of the customer is different in a SPL context compared to the typical role of the customer in an agile team; and also how the use of model-driven development can support the integration between agile methods and SPLE.

Group Discussions

A large number of topics were suggested for group discussions, some of which are:

- What is a "good" architecture for product line creation and maintenance?
- How to reduce upfront domain engineering through agile practices?
- Is there a conflict between long-term domain engineering and short-term customer satisfaction?
- The spit between core asset and application development teams.
- Product lines, by definition, prevent radical innovation and thereby might be potentially harmful.
- What are the tools/techniques needed for agile product lines?
- Exploiting product line platforms for improving/scaling agile processes.
- Who pays for the platform or the core asset development in a product line?
- How to take care of the long-term planning required by product line engineering while still being able to deliver features to the customer in time?

Participants were asked to score the suggested topics. There was enough time for three rounds of discussions; therefore, the top three topics were selected to be discussed. Participants were asked to from three groups – every group was to discuss each of the three topics for 15 minutes and then report on the conclusions of each discussion in 5 minutes. The following is a summary of the discussions.

What is a "good" architecture for product line creation and maintenance?

Basically the question was concerned with the qualities of a good product line architecture that cannot be traded off for agility. That is, if we want to go the agile way about product line engineering, what are the things that we need to keep in mind when it comes to the architecture. Some of the qualities discussed were:

- The architecture should not be too constraining. It should be able to accommodate new features that add business value to the product line – even if these features do not initially fall under the scope of the product line as long as they satisfy the customer's wants.
- It should allow end customers to have their own customizations on existing features without substantial rework.
- It should have minimal dependencies and coupling between components, so that easy plug-in/plug-out of features is possible and cheap.
- A good architecture communicates domain knowledge without the need for extensive documentation. An experience shared by one of the participant suggests that one way to know how well an architecture "communicates" is to ask a

junior developer to describe it.

- It should be able to evolve as new customer requests come in and as market conditions change.
- The adoption barrier of a good architecture should be fairly low. That is, an organization does not need a radical change of course to adopt the architecture – but rather can incrementally move towards it.
- A good architecture has clear metrics of quality that are to be tracked and improved over time.

How to take care of the long-term planning required by product line engineering while still being able to deliver features to the customer in time?

SPLE favors long-term planning activities to speculate what might be needed in the future; whereas agile teams usually plan for the next iteration or next release but try to stay in the near future. The discussion of this topic yielded the following points:

- Based on previous experiences with SPLs, some suggested that you do not necessarily need to go very far in the future. Start with a few customers and try to evolve your product line as more customers come in. Refactoring and regression testing are key to successful evolution.
- Another opinion was that in some cases it might be absolutely necessary to look ahead, but how far you need to look is dependent on a number of factors, some of which are: the stability of the domain and technology trends of the market you are targeting, how well you understand the patterns of the ever changing customer needs in the target sector, and the cultural aspect of planning (e.g. eastern cultures consider long-term planning essential).
- Some reported that understanding 10% of the future requirements was sufficient to maintain the sustainability of their product lines. That is, you do not need to get it all from the beginning.
- Try not to speculate the future, but rather live the future with your customers. One group suggested a scheme they called "The Innovation Wheel." As discussed earlier, get a developer from each product team to form a core asset team. Customers to the core asset team would be: a) product teams that have specific requests to satisfy current needs; and b) a team of market researchers that continuously feed knowledge about what might be a future need. It is important to note that these teams should not be separate but rather interconnected through the alternating roles their members can take from time to time.

Product lines, by definition, prevent radical innovation and thereby might be potentially harmful.

Once you have a product line in operation, you might get too focused on producing what your product line is capable of and not what your organization and personnel are able to achieve. That is, you might put a creativity ceiling that is way below what is practically attainable. Also, especially in the case where a huge investment was put into upfront design and analysis, it becomes too difficult to justify seizing a business opportunity that lies outside the scope of the product line. The discussion of this topic touched on the following points:

- This very idea of blocking innovation comes from the fact that building product lines is motivated by the need to optimize development speed and maintenance – increasing repeatability – which can sometimes be in conflict with innovation and creativity.
- There are examples of product lines that allow innovation, others that support innovation, and those that block innovation. An evaluation of what kind of product line your organization has might be needed.
- The decision to move to a product line practice should not affect the agility to seize business opportunities. It is all about investment decisions. And if we could adopt an agile way to create and manage our product lines incrementally and without large investments upfront, the decision to be innovative becomes easier.
- Experience tells us that being too focused on your current products without discovering new opportunities might eventually drive you out of business. If you do not endorse innovation in the organization, competitors who do will take over.
- It is important to understand that innovation does not necessarily imply that new ideas are beyond the product line scope. Innovation can happen within the product line through improving the capabilities or efficiency of the product line, finding new products that can benefit from the production capabilities of the existing product line, and adding new features as new needs and ideas arise.
- It would be best if you can use the financial benefits the product line practice brings in to the organization to drive innovation beyond the product line.
- It is key to identify the inhibitors of innovation before you take any course of action. It might be that the product line is too constraining, but it might as well be due to other things such as corporate policies, personnel attitudes, or financial constraints. The organization culture can play a vital role in encouraging or blocking innovation. For example, if a radical idea like the iPhone was born in Nokia instead of Apple, would Nokia (with its huge mobile phones product line) have reacted the same way Apple did? We will never know.

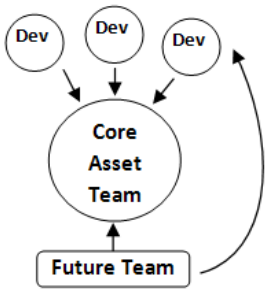

Figure 1 - The Innovation Wheel