Agile software product line engineering: enabling factors

Geir K. Hanssen$^{1,2,*}$

$^1$The Norwegian University of Science and Technology, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
$^2$SINTEF ICT, S. P. Andersens vei 15, NO-7465 Trondheim, Norway

SUMMARY

This paper reports on a study of a software product line organization that has adopted agile software development to address process rigidity and slowing performance. Experience has showed that despite some impediments, this has become a valuable change to both the organization and its development process. The aim of this study is to identify and understand enabling factors of a combined process, and to understand their subsequent effects. Qualitative data are summarized and analyzed, giving insight into the actions taken, their effects that have emerged over time, and the enabling and contextual factors. The study concludes that a combined process is feasible, that the simplified approach makes the organization more flexible and thus capable of serving a volatile market with fast-changing technologies. This has also enabled the organization to collaborate better with external actors. Copyright © 2011 John Wiley & Sons, Ltd.

Received 5 August 2010; Revised 12 November 2010; Accepted 10 February 2011

KEY WORDS: software product line engineering; agile software development; qualitative research; industrial case study

1. INTRODUCTION

This paper reports on a case study of a software product line engineering (SPLE) organization which about five years ago introduced some of the fundamental principles and techniques of agile software development (ASD) to correct a declining (waterfall-like) process performance [1, 2]. Preliminary experiences and studies indicated that this combination of approaches to software product development was both feasible in practical terms and, more importantly, that it to some extent has improved the organizations’ ability to evolve their line of software products, meeting the demands in an increasingly dynamic marketplace [3] with rapid technological shifts. Although being a successful process transformation, it has not happened without impediments, which are related to management of customer engagement [1] and system entropy [4].

Motivated by these first insights, we have done a follow-up study of this organization to understand and describe the agile product line engineering process, how it differs from a traditional product line engineering process and which outcomes this process combination have lead to.

The analysis of the qualitative data contains some implications for practice for other software product line organizations, some explanations of the effects of the combined approach and some pointers for further development and research in the near future.
This report provides novel contributions to the study of agile SPLE as it (1) is a longitudinal study, investigating change over time, (2) is a field study of an industrial case and (3) that it combines knowledge on processes, organization and product.

Although ASD and SPLE share the same ultimate goal, i.e. to enable fast and efficient development of quality software, they represent different approaches to achieving this goal, building on different and even seemingly conflicting values. The main property of SPLE is the planned, prepared and anticipated reuse of a set of artifacts for later efficient constructions of applications. In contrast, ASD according to its original descriptions [5, 6] seeks to avoid anything that does not contribute directly to the current development of software and emphasizes short iterations, the incremental development of working software, and frequent engagement of customers. Simply put, whereas SPLE embodies mainly pro-active and plan-driven development, ASD embodies re-active and change-driven development. Despite these distinct differences and clearly different origins, interest has emerged recently in combining these approaches, which has resulted in an approach known as agile SPLE [7]. There may be a variety of motives for attempting such a combination in an industrial setting, but basically the intention is to improve one approach over the other. ASD methods are mainly concerned with how to organize and drive development at the team level, with short iterations developing a software product incrementally. SPLE gives more advice on high-level and long-termed issues such as variability management and reuse.

2. BACKGROUND

Software product line engineering: A software product line is a set of related products that share not only some common parts; typically software components, but also architecture design, data structures, etc., but still have distinct and different characteristics. The main concept of SPLE is to prepare for and support this anticipated variability in the characteristics of products in the line in order to prepare for reuse and to reduce time to market, to reduce development costs and to improve the quality of software products.

Various comprehensive and detailed descriptions of SPLE practices and principles are available, for example, the frequently quoted books by Clements and Northrop [8] and Pohl et al. [9]. Descriptions vary somewhat in their use of terminology, but at a conceptual level, they describe SPLE as two complementary processes. The first process—domain engineering—deals with the development and maintenance of reusable core or domain assets, such as requirements, design, documentation, code and test artifacts. The second process—application engineering—deals with the development of software products, or applications, using these core assets for rapid and efficient composition of software products. See Figure 1. Beyond this central dual-process concept, various other supportive processes and techniques are described, such as management processes and software architecture practices.

Each of these two main sub-processes is sequential, meaning that practices such as requirements engineering, design, development and testing are ordered in such a way that one activity is supposed to complete before the next can start.

Agile software development: ASD is also a widely known and adopted approach to software development. It differs from SPLE, both conceptually and practically. According to the original idea\(^1\), ASD explicitly seeks to avoid extensive up-front planning, and does not explicitly address issues of reuse. It is an approach that is intended to enable the rapid development of small-scale software solutions from scratch, usually addressing a single, well-defined customer.

There exists a great body of literature explaining the core and peripheral ideas of ASD [10, 11], and even more describing specific agile methods, such as extreme programming (XP) [12], Scrum [6], Crystal Clear [13], Evo [14] and others. In principle, ASD values self-organizing teams, close cooperation with the customer, and informal and direct communication. Documentation, including plans, is kept at a minimum, and work is organized in short iterations developing the software

\(^1\)http://www.agilemanifesto.org/.
product in increments, which is continuously tested in collaboration with customers and potentially refactored. See Figure 2.

External stakeholders such as customers and third-party actors participate in nearly the whole period of a development project. Each iteration, which usually lasts for only a few weeks, starts out by getting feedback from the stakeholders on the results of the previous iteration. This feedback is based on a practical try-out or running demonstration of the product increment. Following this, stakeholders also participate in the prioritizing requirements for the iteration ahead. This also means that requirements management are done iteratively and stands in contrast with the SPLE process where requirements are defined before development.

However, in cases where a software organization serves a large market (like CSoft) this approach to customer interaction is not feasible due to the high number of customers, their diversity and their geographical distribution. Still, applying ASD in such a context needs to find ways to collaborate closely with these customers. Scaling up ASD has emerged as a sub-field of its own [15–17], but the understanding on how to establish efficient customer interaction in large-scale development is still only nascent.
Agile SPLE: Although relatively new, there are already a few reports discussing this idea and potential practical approaches at a conceptual level [7, 18, 19], and a few empirical studies, some of them industry-based [3, 18, 20, 21]. So far, the collective impression from this small body of evidence is that these two apparently opposing approaches to software development may fit together, although not in a straightforward manner. For example, Noor et al. [21] looked specifically at how the product line planning activity can become ‘more agile’. Through a small-scale case study, focusing on the customer collaboration, they identified three criteria for recruiting stakeholders successfully: (i) participants need to be familiar with the fundamental concepts of product line planning; (ii) both external and internal participants need to prepare well and (iii) all participants need to be willing to participate.

Babar et al. [20] report on a case study of a Finnish software product line organization that has been using XP and Scrum for years. Two important lessons may be learned from their study. First, the product development projects need documented background information about the system architecture of the product line and a project team that has tacit knowledge pertaining to and experience of constructing the system architecture. Second, considerable proactive exploratory work is done on the agile development projects before development begins.

The overall impression from the existing studies is that a combined process is desirable because a product line organization needs to both prepare for future development and use the product line as well as improve the performance of developing new releases.

3. RESEARCH CONTEXT

The case organization: CSoft is a medium-sized software company that develops, maintains and markets a product line under the same name. CSoft serves the high-end segment of the market and has a wide international customer base. Despite facing considerable challenges, CSoft is now one of the market leaders; hence, an interesting case to investigate with a potential of providing lessons for other similar organizations. The company was established in 1996 and has grown steadily since then. Currently, it employs about 260 people, including over 60 developers, with offices across Europe, Eurasia and the U.S. There has been a gradual shift from building custom applications to a general software product line. It is important to realize that the SPLE process that is being used at CSoft is not the result of an immediate, ‘big-bang’ adoption and introduction of one of the commonly used process frameworks; rather, it is the result of years of evolution and adjustments.

The product line: The line, or family, consists of several main modules that together support a business value chain of planning, data collection, data analysis and reporting of results (see Figure 3). The modularity allows many configurations and ways to use it. In addition, it offers an application programming interface (API), which is implemented as standard web-services. The composition and use of the modules vary according to customer and case. Some modules can be used in any configuration, while the use of others depends on the situation. The CSoft software also comes with a set of predefined configurations (applications) for the most common usage scenarios and it provides built-in support for detailed customization to create more variants. This means that CSoft does not have separate application engineering projects.

The product line consists of several modules that together support a value chain of planning, data collection, analysis and reporting of results. The composition of a specific product based on the product line can vary in several ways:

Variability in task: Data can be collected in various ways using different modules. For example, there is one module for collecting data via telephone and one for collecting data via the Internet. All variants of the product use the same core component for storing and analyzing data, and

---

1The name is made anonymous due to a non-disclosure agreement between the author and the case organization.

2http://www.w3.org/standards/webofservices/.
have the same functionality for reporting. It is possible to dynamically select features to include according to the license, in order to satisfy the greatly varying needs of the customers.

Variability in application domain: The product line is being used in three different but strongly related domains.

Variability in feature richness: The product is available as in two main variants, the full professional suite and a simple version. The simple version offers only the basic features, is low in cost and is not supported.

Variability in operation: The product is a three-tier application with a web-browser front-end only. This allows for a dual operational model. A customer may buy the product entirely as a service provided by CSoft, meaning that the software runs within a remote hosted environment together with all data, and all access is done through an ordinary web browser (software as a service—SAAS). Alternatively, the total solution can be hosted locally. This is more suitable for customers who use the product extensively and need to retain maximum operational control. However, larger clients use this variant only.

The adoption of ASD: From the start of the company, the development process has matured from a more or less ad hoc type of process (‘creative chaos’) to a well-defined waterfall-inspired process (plan-based and non-iterative). About five years ago, the development process had become too slow and rigid, mostly as a result of the increased size of the organization and complexity of the product line. Out of necessity, CSoft changed to a radically different process: Evo [1, 14], which is an ASD method conceptually comparable to the better-known Scrum method [6], although its terminology differs to some extent.

4. METHOD OF STUDY

4.1. Data sources

All data collected were qualitative and were of the following types:

Semi-structured interviews: eleven interviews were conducted. To form a broad view of the organization and its processes a wide range of roles were interviewed: the architecture team (group interview), the R&D manager, the manager of professional services, the product strategy group manager, product strategy group members, the technical account manager (TAM), a development team leader and a developer. All interviews were recorded and transcribed.

Documents: several documents related to planning and development were collected, for example, all road-map documents (long-term strategic product plans) from one complete release of the product line, related project plans, product fact sheets, presentations given to customers and potential new customers.
Observations: a two-day product conference was used to observe customer presentations, the announcement of new features and strategies, etc. The conference also provided the opportunity to conduct several discussions with a variety of role holders at CSoft as well as their customers, partners and prospects. A customer review meeting was also organized where one of the component teams and a customer team evaluated development.

In addition to these data, which were collected during a period of approximately one year, the analysis and discussion also relate to previous studies, and familiarity with CSoft as reported by the author and others [1–4, 22–24].

4.2. Data analysis

When working with qualitative data, such as interview transcripts, documents and field notes, several strategies are available for analyzing the material. For the case presented in this paper, the preferred approach was constant comparison [25]. In short, textual data are reviewed constantly while they are being collected in order to identify concepts of interest and to investigate these further in upcoming data collection. The growing knowledge of the case is used to plan the collection of new data. Recent findings were typically used in interviews, for example to elaborate on interesting concepts. This continual interplay between the use of information that already has been acquired and information that is being acquired is an example of the hermeneutic circle as described by Klein and Myers [26]—a guiding principle for this explorative study.

When all the data for the case study had been collected, the data were reviewed (by the author) as a whole and an intermediate analysis report was produced. The report (a) summarized process components: roles, activities and artifacts and (b) defined and explained categories or concepts (e.g. ‘community building’ or ‘process simplifications’) that emerged from the analysis of the data. Further, the categories that were identified were analyzed to: (1) find and explain potential internal relationships between data, for example the role of external stakeholders in the product innovation process and (2) relate findings to relevant research, for example to show how findings add to, contrast, or extend previous studies.

4.3. Limitations

This study is subject to the following limitations:

(1) This is a single case study, which naturally affects the generalizability or external validity [27] of the conclusions. Yet there are good reasons for choosing such an approach. First of all, the number of relevant cases is still low. Nevertheless, we had the opportunity to study a case, which we believe is both advanced and relevant to other parts of the software industry. In addition, focusing on a single case means that the study can be more thorough than a study of multiple cases, with respect to available resources. Yin discusses the single case study design [28, p. 38–41] and presents several arguments in favor of choosing such a design. One of these is particularly applicable to CSoft, namely that it is a unique case. According to Yin, such a study may act as a prelude to further studies of a relatively new topic.

(2) A large part of the data that has been collected comes from group and single-respondent interviews with internals and externals. This type of data can potentially be biased, incomplete or even wrongful due to misunderstandings, lack of insight, etc. We have sought to address this threat to internal validity [27] by collecting data from various respondents and supplementing these data with documents and observations.

(3) The third limitation concerns the completeness of the study. Only a subset of the employees was contacted. Likewise, relatively few samples of all available documentation were collected and analyzed. This is due to natural limitations such as limited time and resources.

(4) The fourth limitation concerns the applicability of the findings and conclusions of this study. The organization investigated is a medium-size product line organization and a web/application type of ecosystem (according to the taxonomy proposed by Bosch [29]). Thus, results do not necessarily apply to all other types of software ecosystems.
5. RESULTS AND DISCUSSION

5.1. The agile product line engineering process

The agile SPLE approach at CSoft is described here as the sum of roles, artifacts and processes. The key roles are: module teams of 4–6 developers with an experienced team leader. The teams are fixed and responsible for a dedicated component in the product line. Most teams are related to one or more external stakeholders, which for the most is a customer being invited to participate in the development iterations [2]. A central product strategy group (PSG) consisting of five product managers is the glue between R&D and marketing, sales and operations. This group develops strategic plans as a continuous interplay between nearly all functions in the organization as well as to external actors. Each module (and development team) has an associated product manager. Various other roles support the development teams, especially the chief technical officer (CTO) (having an overall understanding of the total product line) and the architecture team of four experienced developers (having the overall responsibility of improving the total system architecture and to hinder system entropy).

The main artifacts that support the process are the product roadmaps, which state long-term (1–2 years ahead) strategic objectives for the product line. These are developed by the PSG, one per module. The project plan is developed in collaboration between the PSG and the development teams and defines scope and design ideas of the next release. Typical SPLE artifacts, such as requirements documents, architecture documentation, variability models or domain models, are produced.

Three process levels [3]: A strategic process focuses on the development of the roadmaps, which is the result of a continuous dialogue with internal and external actors. This process needs to encompass technological developments, domain development and ideas and needs from various customers. The strategic process covers most of the domain engineering tasks defined by SPLE process frameworks [9], however with a minimum set of output. For example, it does not produce a domain or an architecture model. The only outcome is a set of product roadmaps, one per main module of the product line.

A tactical process consists of the development projects. Each development team runs an agile Evo-project with fortnightly iterations, equivalent to the sprints in Scrum. This process is organized according to the principles described in Figure 2. Working software is deployed on test servers at the end of every iteration and the representative stakeholders then evaluate the latest results and give corrective feedback to the development teams [2, 3]. All development projects (one per module) lasts for approximately one year and results in a new main release. If needed, intermediate releases with shorter lead time may be initiated.

An operational process covers day-to-day activities such as support, training, key account management, etc. Information such as error logs, support requests and general feedback goes into the strategic process. However, in cases of urgent errors or problems, hot fixes are made to correct the problem as soon as practically possible. Such corrections do not imply a change in functionality and are easily deployed to most customers as the solution is operated as a service. Customers that operate their own installations naturally need to update the system locally (Figure 4).

5.2. Enablers of the process combination

By comparing the processes and the organization at CSoft with what can be referred to as traditional SPLE we see that CSoft has taken two major steps to reshape their development process: (i) the overall simplification of processes and organization and (ii) the close involvement of customers, in both planning and development.

Simplification of process and organization: (1) No separation between core asset development and product development: The idea of having two distinct processes in SPLE is to enable the separation of the different concerns of building a robust platform on the one hand, and of effectively building products based on that platform on the other. At CSoft the product line is developed and released as a single unit and the total organization manages to handle both the development of core assets and applications simultaneously [30]. This is an important principle with nearly no
exceptions. A member of the architecture team explains:

We have a strict policy that we usually do not make things for single customers. However we sometimes do that, but then the experience often turns out to be that we should not have done it. It has a high cost and low reuse value.

This is done as simple as possible; having fixed development teams that create the next release (corresponding to application engineering) and the PSG that develops the roadmaps (corresponding to domain engineering). CSoft is not doing domain engineering as a separate or dedicated process in the sense that they produce domain artifacts that are to be used later in a separate application engineering process. Artifacts are developed based on identified needs and ideas coming from the roadmaps, but with constant corrective feedback from agile development iterations. Thus, this is a case of re-active domain engineering, as opposed to pro-active approach.

The R&D manager explains:

We [R&D department] are now organized so that we have product teams that work on the various modules of our software. Each team has a team leader. Correspondingly on the PSG side we have a product manager, which faces the market, he does some sales work. He is the one that know the road map for his module best. Those two together are responsible of delivering a roadmap before we decide the content of the next release.

(2) Simplification of requirements management: CSoft only documents requirements in product roadmaps (at a high level), which are long-term strategic plans for the product line and project plans, covering the upcoming release project with nearly a year work-time. As the projects get started, the two-weekly iterations will manage requirements according to the principles of ASD. Each iteration has a planning meeting, in which the prioritized requirements for the next release are detailed. See Figure 5. Thus, requirements management becomes an ongoing process. This is in sharp contrast to the lengthy process illustrated in Figure 1.

(3) Direct communication and coordination over formal documentation: The general principle of openness, visibility, and direct communication is adopted from ASD. This means that the status of the development of roadmaps and the status of development as it progresses through the iterations are visible to everyone in the organization as well as customers functioning as external stakeholders. Direct, face-to-face, communication is emphasized and written communication is
kept to a minimum, giving efficient communication [31]. As an example, one of the developers describes the meeting with the external stakeholder at the end of the two-week Evo iteration:

*What you get during a meeting is often very valuable. Especially when you are about to move in the wrong direction, which you can adjust. We get feedback saying that our solution is not quite what they had in mind or what they need.*

**Engagement of customers:** Interacting with customers is an obvious but nevertheless critical factor for success in any type of software engineering [32, 33], SPLE included—although traditional SPLE frameworks tend to be waterfall oriented. Typical objectives when interacting with the customer are to define short- and long-term requirements, gain access to domain and business knowledge, verify and validate results and even in some cases involve customers as innovators [34]. Collaborating wisely may yield several benefits for software development, such as improved system quality due to more accurate user requirements, the avoidance of costly system features that the user does not want or cannot use, improved levels of acceptance of the system and better understanding of the system by the user resulting in more effective use [35].

CSoft has a continuous, wide and very rich communication with direct and indirect users of their product line. This covers the short-term development projects, which have invited customers as stakeholders in most iterations and long-term strategic planning where many roles in the organization have a continuous dialogue with customers, for example, through sales, support and TAM.

From our observation of one of the customer review meetings we saw a lot of very detailed discussions that were made possible by on-the-fly demonstration of the software through the screen sharing solution. This sparked detailed discussions both on the customer side and among the development team. The meeting resulted in a list of clear actions points to be addressed in the next development iteration.

This is an example of what Keil and Carmel define as direct ‘customer-developer’ links [36]. They show that direct communication between the supplier and a participating customer reduces problems with filtering and distortion. The more links that are used, the better the communication becomes, up to a certain point.

Although the close relationships with customers have shown to be vital at CSoft, they might nonetheless be fragile and there is a clear need to manage the many inputs received from, and the expectations of, multiple external actors. This includes the need to continuously maintain a motivation to participate [1] and to balance radical and incremental innovations [37, 38], avoiding

---

**Table 1:** The ASD iteration plan.

<table>
<thead>
<tr>
<th>Week</th>
<th>Days</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Monday</td>
<td>Technical meeting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Step kick off</td>
</tr>
<tr>
<td>2</td>
<td>Tuesday</td>
<td>Development</td>
</tr>
<tr>
<td>3</td>
<td>Wednesday</td>
<td>Internal demo (prepare for meeting)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Evo step planning and review meeting</td>
</tr>
<tr>
<td>4</td>
<td>Thursday</td>
<td>Retrospective</td>
</tr>
<tr>
<td>5</td>
<td>Friday</td>
<td>Project Status Meeting</td>
</tr>
</tbody>
</table>

---
innovation lock-in [39]. The manager of the PSG group explains:

Everybody has busy jobs and projects that need to be on time, etc. It happens quite often that we have to cancel these meetings or that they haven’t done anything since the last time. Then we can only show them what we’ve done and get some ad-hoc feedback...

Domain volatility: One assumption about SPL organizations and their target domains is that the domain is fairly stable [19]. Serving a domain that does not change too much means that there is a lower risk in pre-developing assets for later (re)use in application engineering. In contrast, the domain that CSoft serves is unstable, because its boundaries are still being determined. New technologies and business ideas emerge frequently; hence, it is more strategically important to be able to respond quickly to the market than to produce reusable components that might be useful at a later moment, but are not sure to be. CSoft have adapted to this volatility by having several and frequent links to customers. One of the TAM explains:

We [technical account management] are the people dealing with the customer on the day to day basis. And you will find the engineering guys deal with them during the development cycle on a week-to-week or fourth night to fourth night type of situation...

The SAAS delivery model: Early in the development of the CSoft product line, the SAAS deployment model [40] was adopted. Offering a purely web-based solution is part of the key business idea. When a new release is ready, all parts of the product line are released simultaneously and deployed on a server farm, which allows customers to use the new release instantly through an ordinary web browser. This model ensures easy and rapid release of new versions of the product line, offers an easy upgrade path for the customers and provides opportunities for reducing the costs because no local operational infrastructure and services are needed. Actually, as the customer under a SAAS regime does not own anything but data and a license to access services and some basic infrastructure, the traditional concept and concern of TCO get simplified. In addition, and most relevant to this study, the SAAS model is advantageous for the development of new releases. Owing to the agile (Evo) iterations, an external stakeholder will normally participate in the preparations of an iteration by specifying needs and requirements. Later on, when the outcome of the iteration is to be evaluated, it is practical for the external stakeholder to access and test out the latest increment, because it is deployed on a dedicated test server and immediately accessible as a service, just like the stakeholder is normally used to. A short extract from the transcript of a customer review meeting may serve as an example of the mutual value of offering the software under development as a service in the development process (on a SAAS test server).

(Product manager have been demonstrating new features live on an SAAS test server through distributed screen sharing.)

Product manager: ‘Just before we conclude; would you like access to the test servers?’
Customer representative: ‘Of course…we can play with the new version.’
Product manager: ‘Would you be interested in a follow-up meeting?’
Customer representative: ‘Yes yes.’
Product manager: ‘That’s good, then I will contact you.’

5.3. Experience with the combined process

Risk reduction: Developing and improving a software product line may carry with it a set of risks that need to be managed. McGregor describes risks that are inherent to SPL, depending on whether a proactive or reactive strategy is being applied [19]. With a proactive strategy, where assets are predeveloped with the assumption that they will be used in future applications, there is a risk that these assets may become obsolete, thereby becoming a lost investment of resources. With a reactive strategy where assets are harvested from applications for later reuse, there is a risk that short-term business opportunities will be missed. There is also a risk that a lot of rework will need to be done to prepare such assets for future and more generalized reuse. CSofts
approach, or strategy for managing their product line, is best defined as incremental. Further, as McGregor describes, it constitutes a compromise between the proactive and reactive approaches, which reduces the risks mentioned above. Given that all development is aimed toward the next release, which will be a maximum of one year ahead, there is a low risk of developing features that will not be used. This is actually a central motivation for using the Evo process \[14\], which is driven by short-term goals. Another observed effect of the agile approach to requirements management is that frequency and closeness increase the ability to catch both explicit and implicit (tacit) requirements \[41\].

One of the product managers explained the importance of frequent feedback through customer involvement, both in strategic planning and in the actual development:

*Customer involvement, or stakeholder involvement is crucial to ensure that we get information early on in the roadmap planning to understand what people out there need [...]. And next as we do the work we have close involvement with them so that they can see what’s going on, they can give us feedback still in the development stage so that they can say ‘actually it’s good but if you’d do it like this it would be slightly better’, or ‘have you thought about this kind of consideration here?’*

*Reduced maintainability:* The present situation at CSoft has emerged after several releases, as the organization has prioritized to serve the market rapidly with new and improved features and functional qualities at the expense of internal ‘tidiness’. This may have been a wise strategy in the short term, because it has helped the organization to establish its position as one of the market leaders, with a product line that generates enough income to support further (sustainable) development. However, this is more or less a deliberate strategy for the shaping of the organization and the product line has come at a cost of escalating system entropy, which now makes it more and more difficult to improve the product line. Through a recent study of the maintainability of CSofts product line architecture \[4, 22\], we found that R&D experiences severe problems being potentially enforced by an ASD approach. It is very difficult for even experienced developers to understand the inner structure and workings of the core parts of the product line, especially in components that are central in most application compositions (see Figure 3—The product line structure). This has a negative effect on the developers’ ability to modify and develop the product line. A member of the architecture team explained:

*...people [developers] are afraid to change code. I talked to one of the developers I have helped, I said that ‘why not change that piece?’ and the answer I got was that ‘no, this is mission critical code’. This code was so critical that he did not dare to touch it.*

If this situation is not managed adequately, it may seriously threaten the positive effects of agile SPLE. As a result, an extensive refactoring of the complete product line architecture has recently been initiated. Also, as Ghanam and Maurer suggest \[42\], building and maintaining acceptance tests defined by business stakeholders may be an important approach to manage complexity without sacrificing agility.

*Community building:* Following the growth of the product line and the extensibility of it through an API, a third-party community has emerged. This group of nearly 60 external organizations uses the product line as a platform to produce related services and products. This ‘extension’ of the organization holds a great business potential to CSoft \[43\]. First, it allows them to maintain maximum focus on the development and progression of the core product line. This is extremely important, because if the level of complexity becomes too high (from additional solutions), their ability to improve the core product line would be seriously hampered \[44\]. Second, the community increases the actual value of the product line to the customers, increasing sales, and increasing the ‘stickiness’ of the platform. This platform-based community resembles somewhat open source initiatives, but with the big difference that CSoft is the leading organization, being in control of the platform. In order to enable the third-party community to direct their efforts optimally, CSoft share their high-level plans openly and, in some cases, invite third parties to planning and development meetings, just as they do with customers.
We spoke with several representatives from external organizations during a product conference. For example, one told about a case where his company gave input to CSoft of some changes they would have liked to see. This led to CSoft inviting a delegation from this company to R&D in Oslo. Spending a whole day with 10 CSoft people, discussing the solution with four from this company. This way of being actively included was perceived very positive and actually affected the software in the end, to the benefit of this external organization.

This nearly symbiotic relationship between such actors in the third-party community and CSoft as a product line organization and a provider of a partially open platform can be described as a software ecosystem [45, 46]. This is an emerging concept within software engineering and a potentially important shift in how the industry relates to other actors in the market. A recent and related study report [47] develops this concept further, relating it to the concept of ‘organizational ecology’ coming from the socio-technical theory [48].

Openness and visibility: Through the lifetime of the organization there has been a shift in how CSoft deals with information, from keeping information internal toward more and more sharing of strategies, plans, results and interfaces externally. This openness can be seen in several practices:

1. Customers are participating as stakeholders in development projects. As an example, one of the developers explained his experience with talking directly with customers participating in development:

   The feedback you get from those meetings are valuable—especially when you are about to move in the wrong direction. It means that we can make adjustments—that they [customers] can tell us that our solution is not what they have in mind or what they need. We can discover misunderstandings or misinterpretations. We try to solve problems in a way that makes sense to us, but when actual users of our solution tries it out we see that we have misunderstood the process, for example.

2. A couple of years ago a large product conference was organized to inform about the latest developments and plans and to have formal and informal discussions with customers and actors from the third-party community. The goal was to gather actors having a relationship with CSoft to openly share information and experience. The final statement from the CEO in his opening speech was:

   Network with your peers, with our exhibitors and with your key vendor contacts. Learn best practices from other customers, from industry experts and from CSoft product experts.

3. Each year an advisory board of some of the largest customers meet with the PSG and others to discuss needs, ideas and future development. (4) On an ad hoc basis customers or third-parties with good ideas or interesting needs visit one or more of the module teams. All in all, there is a continuous dialogue with external actors at many levels—an important part of the strategic process. This way of interacting with customers and third parties resembles what Chesbrough [49] has termed open innovation, an approach in which innovation takes place partly across organizational boundaries, rather than within them, which may be more common in traditional product line organizations.

   Enkel et al. present open innovation as consisting of one of the three core processes [50]. The first, which is called the outside-in process, enriches the company’s own knowledge base through the integration of supplier, customers and external knowledge sourcing. The second is called the inside-out process, in which ideas are brought to the market, which may generate new streams of income through spin-offs or joint ventures. The third, which is called the coupled process, combines the first two and describes CSofts approach. This openness across organizational borders and the ability to manage both internal and external innovation processes may constitute a clear competitive advantage [51] and, in the case of CSoft, it has enabled the organization to improve the processes by which their software product line continues to develop.
6. CONCLUSIONS

The aim of this study has been to identify and understand enabling factors of a combined process of ASD and SPLE, and to understand the effects of it. On the basis of the collected data and the analysis we have done, we draw the following conclusions:

(1) We confirm what others have reported; it is certainly feasible to combine ASD and SPLE. The main contribution from the agile side is simplification of the development process. Compared with established SPLE frameworks, a more agile-oriented variant can reduce the costs of developing intermediate assets that do not contribute directly (short term) to the product line. In agile terms, this is called ‘maximizing the value of work not done’.

(2) Managing a relatively simplistic process also makes it a changeable process, which is important to an organization that is still developing a rapidly changing technology and adapting to a turbulent market.

(3) The close and continuous engagement of externals makes the organization responsive to detailed input from the market, both throughout development and in strategic planning, thereby strengthening innovation capabilities. Supporting an external community is the key to maintain focus on the evolution of the core product line and to drive innovation.

(4) We believe that an agile SPLE approach is particularly suitable to an organization that serves a volatile market. The ability to respond rapidly to new opportunities that are provided by the constant development of new technology and the continual shaping of the market is of the utmost importance if the organization and its products are to mature. Also, the deployment model (SAAS) contributes to the simplification of the SPLE process.

(5) The combined process can enable and drive development of the organization itself. However, and as a warning to others, the agility of the SPLE process can aggravate problems of system entropy and poor maintainability, thus calling for corrective actions.

Implications for practice

(1) Support the external environment by being open. Sharing information about plans, strategies and development may create a fundamental base for collaboration and new patterns of innovation. If appropriate, encouraging and supporting a third party community can be a valuable extension to the normal development of the product line.

(2) Reducing the variability in the product line consequently reduces the costs of maintaining and managing variability. The potential drawback of reduced variability can be compensated through offering interfaces for extending the product line by externals, and thus increasing the overall variability.

(3) Establishing and benefiting from an SPLE/ASD-combination takes time. A successful development relies on repeated cycles of experimentation and learning, affecting organization, product line and processes. This learning process needs to involve all types of actors.

These conclusions and implications for practice should motivate and justify further research on the potential benefits and challenges of utilizing ASD principles in SPLE. Directions for further research would seem to be twofold: (1) to conduct further industrial case studies and surveys to build empirically based knowledge; and (2) to conduct more studies on the interaction between the product line organization and its external environment such as customers, third parties and other external factors.

SPLE and ASD have originally different home grounds. Historically, their differences have been meaningful but as the industry now develops toward more open processes, distributed and inclusive development of technology, we believe that these two approaches will benefit if combined. As a main conclusion, we see that the impact of applying some of the fundamental principles from ASD in an SPLE organization is that it may (in this case at least) produce a more flexible development process, which is more open toward the organizations external environment, enabling the organization to benefit from open innovation.
ACKNOWLEDGEMENTS

The authors are grateful to the people at CSoft for sharing their precious time, providing highly valuable input to this study. This work was done as part of the EVISOFT project (grant 174390/140) and the Agile project (grant 179851/140), both partly funded by the Research Council of Norway.

REFERENCES


