The Journal of Systems and Software 85 (2012) 1455–1466

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A longitudinal case study of an emerging software ecosystem: Implications for practice and theory

Geir K. Hanssen[a,b,*]

[a] Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), Sem Sælands vei 7-9, NO7491 Trondheim, Norway
[b] SINTEF ICT, NO7465 Trondheim, Norway

ARTICLE INFO

Article history:
Received 6 May 2010
Received in revised form 6 April 2011
Accepted 7 April 2011
Available online 19 April 2011

Keywords:
Software ecosystems
Software product line engineering
Agile software development
Longitudinal case study

ABSTRACT

Software ecosystems is an emerging trend within the software industry, implying a shift from closed organizations and processes towards open structures, where actors external to the software development organization are becoming increasingly involved in development. This forms an ecosystem of organizations that are related through the shared interest in a software product, leading to new opportunities and new challenges to the industry and its organizational environment. To understand why and how this change occurs, we have followed the development of a software product line organization for a period of approximately five years. We have studied their change from a waterfall-like approach, via agile software product line engineering, towards an emerging software ecosystem. We discuss implications for practice, and propose a nascent theory on software ecosystems. We conclude that the observed change has led to an increase in collaboration across (previously closed) organizational borders, and to the development of a shared value consisting of two components: the technology (the product line, as an extensible platform), and the business domain it supports. Opening up both the technical interface of the product and the organizational interfaces are key enablers of such a change.

1. Introduction

A recent development within software engineering is the emergence of *software ecosystems* (Messerschmitt and Szyperski, 2003; Bosch, 2009). This new concept and its implicit reference to ecology imply a shift of focus from the internals of the software organization (the individual organism) towards its environment and the relations and actions within (the ecosystem). Viewing the software industry and the market it serves as an ecosystem, may introduce a set of new challenges and opportunities (Jansen et al., 2009a,b), for example new business models, open innovation, collaborative development, issues of ownership, strategic planning, and variability management. This seems to be a part of a general development in the software industry (Qualman, 2009), where customers expect to be more involved in the shaping of the technology they use, where innovation is no longer an internal matter, where time to market is decreasing, and adoption rates are increasing.

To understand some of this ongoing development we have studied CSoft, a medium size software product line organization for a period of approximately five years. During this time they have moved from a closed and plan-driven approach to a practice of agile software product line engineering (SPLE) and now further towards a software ecosystem. Based on three recent studies of this organizational development we have designed and conducted a final study, collecting new qualitative data to investigate in more detail this development in general, and how the organization relates to its environment in particular. Inspired by our detailed insight into a real industrial case and with the ambition to try to understand some of the details of how a concrete ecosystem develops, we have define research question 1 to be:

Why and how is software product line engineering developing towards a software ecosystem?

Our study will give insight into three aspects of this question: (1) how this ecosystem has emerged, (2) how the present organization works in terms of its structure, its processes, and its product line, and (3) how the organization relates to actors in its external environment.

To help develop this understanding and to contribute to the growing research on software ecosystems, we relate our findings from this case study to the general theory of organizational ecology (Emery and Trist, 1965; Trist, 1977), which is derived from sociotechnical theory. This theoretical platform has matured for decades and describes how organizations in general relate to their external

* Correspondence address: SINTEF ICT, box 4760 Sluppen, NO-7465 Trondheim, Norway. Tel.: +47 92492454; fax: +47 73592977.
E-mail address: ghanssen@sintef.no

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.04.020

environment. We have found that this knowledge is relevant, applicable and beneficial to the software engineering domain, and that existing literature on software ecosystems lacks this theoretical connection. Thus, research question 2 is:

What are the characteristics of software ecosystems?

Several recent studies have suggested definitions and developed important concepts but the terminology and connectivity between these concepts are still vague. We believe that this field of research can benefit from the development of an empirically grounded theory of software ecosystems. The results obtained from this study are used to propose such a theory, as a starting point. We also discuss implications for practice as well as providing advice for further research.

The rest of this paper is organized as follows: Section 2 provides relevant background information about the concepts being discussed, such as socio-technical theory, organizational ecology and software ecosystems. Section 3 explains the applied study method. Section 4 provides the results of the study as a thematically structured overview of the qualitative data, illustrated by quotations from interviews and observations. Section 5 presents a discussion of the results related to the defined research questions and provides implications for theory and practice. Section 6 concludes the study and provides directions for future research.

2. Background

2.1. Socio-technical theory and organizational ecology

Socio-technical theory is a view of the organization as the sum of and interplay between a social system and a technical system. That is (1) the people, their relations, their knowledge, and how they work together as a whole, and (2) the tools and techniques being used to perform the work. A fundamental principle of socio-technical theory is the natural interdependence between these two systems, meaning that to improve the performance of the organization (productivity, quality of work, etc.) both subsystems have to be considered at the same time. Changing one affects the other. This principle was first drafted by Trist and Bamforth (1951), based on their studies of changes of work processes in coalmines.

Through studies that followed, the theory has been developed to consider how organizations relate to their external environment (Emery and Trist, 1965). This implies that an organization has to be understood as both (1) the internal interplay between a social subsystem and a technical subsystem (the socio-technical system), and (2) the interplay between the organization and its external environment.

Emery and Trist (1965) developed a simple classification system of four types of organizational environments – forming a series, in which the degree of causal texturing is increased. Thus, understanding and ordering the types of environments is useful in understanding socio-technical systems beyond the limits of a single organization. The first, and the simplest type, is 'the placid, randomized environment' where "goods" and "bads" are unchanging, and are randomly distributed in the environment (Emery and Trist, 1965, p. 7). The optimal strategy is to 'do one's best' on a purely local basis – there is no difference between strategy (planning) and tactics (execution). The second type is 'the placid, clustered environment' where "goods" and "bads" are not randomly distributed but band together in certain ways. Strategy is different from tactics, and survival becomes critically linked with what an organization knows about its environment. Organizations in this environment tend to become hierarchical, with a tendency towards centralized control and coordination (Emery and Trist, 1965, p. 8).

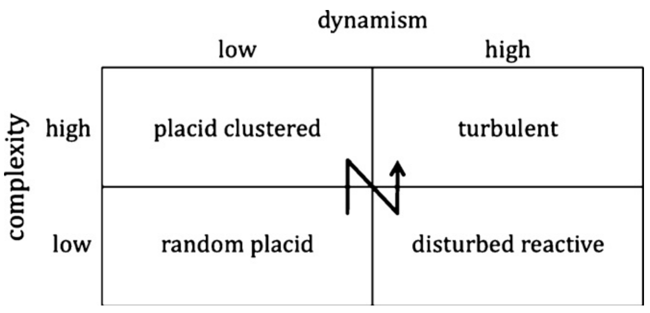

Fig. 1. Types of organizational environments.

The third type, 'the disturbed-reactive environment', is an environment where there is more than one organization of the same kind. The existence of a number of similar organizations becomes the dominant characteristic of the environmental field. These organizations compete, and their tactics, operations and strategy are distinguished. The flexibility encourages a certain decentralization, and it also puts a premium on quality and speed of decisions at various peripheral points (Emery and Trist, 1965, p. 9). The fourth, and the most recent type is 'turbulent fields'. This type implies that significant variances arise from the field itself, not simply from the interaction of the component organizations. Three trends contribute to the emergence of these dynamic field forces: (i) the growth to meet type-three conditions, (ii) the deepening interdependence between the economic and other facets of the society, and (iii) the increasing reliance on research and development to achieve the capacity to meet competitive challenge. A change gradient is continuously present in the field (Emery and Trist, 1965, p. 10) (Fig. 1).

This interplay, inherent in turbulent organizational environments, has been further studied, leading to the development of the concept of organizational ecology (Trist, 1977). It is particularly relevant to organizations operating in complex and unstable domains. Viewing the organizational environment as an ecosystem means that it is considered to be an open system as opposed to a closed one, organizational borders are permeable, and organizations relate dynamically to other organizations in the same field.

Developing the ecology concept further, Trist describes three classes of organizations within the turbulent environment. In a Class 1 system member organizations are linked to a key organization among them. The key organization acts as a central referent organization, doing so even though many of them are only partially under its control, or linked to it only through interface relations. Interface relations are as basic to systems of organizational ecology as superior-subordinate relations are to bureaucratic organizations. Interface relations require negotiation as distinct from compliance. In a Class 2 system the referent organization is of a different kind. It is a new organization brought into being and controlled by the member organizations rather than being one of the key constituents. A Class 3 system has no referent organization at all.

Technocratic bureaucracies have been the natural organizational form for disturbed-reactive environments (up to the 1960s or so), and this is a form that has been applied to many software engineering organizations. However, this type of system fails to adapt to conditions of persistent and pervasive environmental turbulence, mostly because it is constructed and optimized to work well in stable environments. This leads to the emergence of the new ecologically oriented systems, which show clear differences in that they promote self-regulation (as opposed to centralized control), and that they have a networked character (as opposed to segregated organizations). According to Trist (1977, p. 172), such systems, lacking formal structure, exist through the use of technology. Further, they also need shared values. Trist used the example of the 60/70s youth-culture that had a set of distinct (political) values. A more appropriate example from a business perspective is a shared value in growth and profit.

Unlike the micro-level (the single organization) and the macro-level systems (society), the intermediate level systems (organizational ecosystems) are hard to see, understand, and describe due to their weak structuring. They are also the most recent type, so there is less experience with them. This relates especially to software engineering ecosystems, which is a new but rapidly advancing concept (Messerschmitt and Szyperski, 2003; Bosch, 2009) despite decades of development of software engineering as a practice and business. This approach is driven by the Internet as a rich and speedy collaborative platform (the technology), and a common interest in the product line (the shared value).

2.2. Software ecosystems

Software ecosystems is a more recent term, that refers to a networked community of organizations, which base their relations to each other on a common interest in a central software technology. Some other definitions of this emerging concept have been proposed, for example by Messerschmitt and Szyperski (2003): "*a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them.*" (p. 2). Another definition by Bosch (2009), focusing more on the common interest in the software and its use, is: "*the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions.*" (p. 2).

Well-known examples of communities that may be seen as software ecosystems are Apples iPhone/Appstore platform, and the open-source development environment Eclipse. The first is an example of a partially closed and controlled ecosystem, and the latter is an example of an open ecosystem allowing more flexibility in use and development. This simply illustrates that the ecosystem concept may refer to a wide range of configurations. Yet, they all involve two fundamental concepts: (1) a network of organizations and (2) a common interest in central software technology. These organizations may have different relations to the central software technology, and for this reason, different roles in the ecosystem. There are *three key role types*:

- First, one organization (or a small group) acts as the *keystone organization*, and is in some way leading the development of the central software technology.
- The second key organizational role is the *end-users* of the central technology, who need it to carry out their business, whatever that might be.
- The third key role is *third party organizations* that use the central technology as a platform for producing related solutions or services. In addition to these key roles, various other related roles might be part of the ecosystem (Jansen et al., 2009a), for example standardization organizations, resellers, and operators.

A fundamental property of the central software technology is that it is *extensible beyond the keystone organization* (Alspaugh et al., 2009). Extensibility can be achieved in various ways, for example by providing an application programming interface (API) or a software development kit (SDK), by supporting exchange of open data formats, or by offering parts of the technology as open source. Opening up the technology in these, and potentially other ways, enables external organizations to use the central software technology as a *platform* where existing services or data can be used and extended. Bosch (2009) proposed a Software Ecosystem Taxonomy that identifies nine potential classes of the central software technology, according to classification within two dimensions. The first one is *the category dimension*, which is ranging from operating system to application, and to end-user programming. The second one is *the platform dimension*, ranging from desktop to web, and to mobile. The case discussed in this paper is an application-web type.

The keystone organization has a special position in the ecosystem as it controls, strictly or loosely, the evolution of the central software technology. This may include various responsibilities, for example typical software product development activities such as strategic planning, R&D, and operational support. These responsibilities come in addition to activities specific to ecosystems such as enabling efficient external extensibility, provision of insight into planning and development, and supporting ecosystem partners in various other ways.

One potential benefit of being a member of a software ecosystem is the opportunity to exploit open innovation (Chesbrough, 2006), an approach derived from open source software processes where actors openly collaborate to achieve local and global benefits. External actors and the effort they put into the ecosystem may result in innovations being beneficial not only to themselves (and their clients) but also to the keystone organization, as this may be a very efficient way of extending and improving the central software technology as well as increasing the number of users. Closer relationships between the keystone organization and the other actors may drive both an outside-in process as well as an inside-out process, as described by Enkel et al. (2009). Also, the proximity between the organizations in an ecosystem may enable active engagement of various stakeholders in the development of the central software technology (Hanssen and Fægri, 2008).

The ultimate objective for investing in and working towards an ecosystem is that all members will gain more benefits from being a part of it, as compared to the more traditional approach for software product development with segregated roles, a low level of collaboration, and closed processes. A well functioning ecosystem is, in summary, a complex configuration with collaboration across traditionally closed organizational borders. Such multi-organizations are probably not established as a deliberate, planned effort. Rather, they emerge as a result of many congruent factors such as technology development, globalization, new collaborative patterns, and clients becoming more and more accustomed to participating in the shaping of the technology they use.

3. Study method

The case study reported in this paper is the last in a series of four consecutive studies of the software product line organization CSoft, constituting a longitudinal study started in 2004, and now covering five years of the organization's history (Hanssen and Fægri, 2006; Fægri and Hanssen, 2007; Hanssen and Fægri, 2008; Hanssen et al., 2010). In addition, two earlier papers written by other authors provide background information on the historical development of the organization (Moe et al., 2002; Johansen, 2005). Together, this provides a valuable insight into the longitudinal development of a software product line organization.

The name of the case organization and its product is kept anonymous due to a non-disclosure agreement that has been signed by the author.

3.1. Study type

The study can be classified as a longitudinal single case study. We have applied a set of principles for interpretative field studies defined by Klein and Myers (1999) (Table 1).

Table 1
Application of Klein and Myers seven principles of interpretive field research.

Principles (from Klein and Myers, 1999, p. 72)	Practiced in the case study
1. The Principle of the Hermeneutic Circle	Data are collected through repeated interviews with actors playing various roles. The data collection is supported by observations and as collection of relevant documentation. The growing knowledge of the case has guided the data collection.
2. The Principle of Contextualization	The study of the case is conducted from two viewpoints – the present organization and its activities, and how this organization has emerged over time.
3. The Principle of Interaction Between the Researchers and the Subjects	A large part of the collected data is based on semi-structured interviews (Seaman, 1999) that followed open interview guidelines to ensure a balance between thematic focus and room for reflection, correction, and discussions. This allows for unplanned but relevant topics to be addressed.
4. The Principle of Abstraction and Generalization	Findings are related to the concept of organizational ecology (Trist, 1977), derived from socio-technical theory. Key principles from this theoretical background are applied to the studied case. Some are adopted, some are adjusted, and some are added.
5. The Principle of Dialogical Reasoning	The theory applied to the case was *not* used to plan and guide the data collection. The applicability of the theory became evident through the analysis after the data had been collected.
6. The Principle of Multiple Interpretations	This principle is followed by collecting data from both external actors and people with various roles in the product line organization.
7. The Principle of Suspicion	The data have been collected and analyzed by the author, who is external to the organization, having no formal responsibilities, interests or agenda, except to create an unbiased view of the organization and its development.

3.2. *Data sources*

During 2008, 2009 and 2010 new data were collected to investigate the agile software product line organization, it s processes, and in particular how they relate to external actors. Data are of three types: interviews, observations, and collected documents. Table 2 shows the list of content for each type.

3.3. *Sampling and collection*

The focus of this study has been to investigate how CSoft relates to external actors such as customers and third party organizations. This has guided the sampling of interview respondents, selection of events for observation and documents to be collected. Interview respondents have been asked to recommend other respondents, based on their understanding of the study (snowball sampling). A single-person interview lasted approximately 30–40 min. Group interviews lasted up to 3 h.

All data have been collected and stored in a database for later analysis. Interviews were recorded using a digital voice recorder and then transcribed.

3.4. *Analysis*

Data have been analyzed in two steps:

Step 1 – All data were first examined to produce an intermediate analysis report, which documents the development process in terms of roles, activities, and artifacts, in addition to high-level concepts, necessary to understand how product planning and development is conducted. This analysis created a structure by grouping information coming from the various data sources. Examples of such concepts are teamwork, planning, and innovation. The objective of this report was to establish a broad understanding of the context, i.e. the organizational set-up and its processes. The report has been used in the description of the study context (3.5), as well as a preparation for step 2.
Step 2 – All data, in textual format, were analyzed using a tool for qualitative data analysis, NVivo^{TM}. Data were coded, meaning that fragments of text, for example statements, facts, comments, concerns, and ideas, were tagged with nodes describing the data fragment. Examples of such nodes, which emerged from the analysis, are '*co-creation*', '*finding the right stakeholders*', '*learning of business processes and domain*', etc. These are detailed in Section 4. This way of analyzing the data develops a meaning and an interpretation of the data, and relates fragments from different locations in the data material to concepts, which may be grouped into categories. These results can then be used as the basis for formulating a theory explaining some of the findings. This approach resembles 'grounded theory' in that a theory is developed, and that it is grounded on data (Glaser and Strauss, 1967). The theory being developed may be new, but it can also be

Table 2
Collected data.

Interviews	R&D manager (semi-structured interview) Manager of Professional Services (semi-structured interview) Product Strategy Group manager (semi-structured interview) Product Strategy Group members (3 semi-structured interviews) Technical Account Manager (semi-structured interview) Team leader (semi-structured interview) Team member/developer (semi-structured interview) Product Strategy Group manager (follow-up interview after observation of the review meeting) 2 (of 3) members from the Architecture Team (group interview)
Observations	Product conference (various presentations and ad hoc conversations) Customer review meeting (one R&D team + customer team + sales) Webinar presentation of the new API
Documents	Component A–E project roadmaps Chief Strategy Officers keynote at a product conference Chief Executive Officers keynote at a product conference Vice President Product marketing – presentation at a product conference Customer's presentation at a product conference

related to an already established theory. In the case of the CSoft study, the analysis is related to the organizational ecology concept explained in Section 2.1, and it seeks to apply this theory in a software product-engineering context. Implications for theory are discussed in Section 5.1.

3.5. Study context

3.5.1. The organization, processes and the product line

The organization. CSoft is a medium-size software company that develops, maintains, and markets a single product line under the same name. They have now become the market leader in the high-end segment of the market. Currently CSoft employs about 260 people, including more than 60 developers. The main office is located in Oslo, which houses the main section of the development department as well as top management and various support services such as operations, technical support, sales, training and others. The rest of the organization is distributed internationally with development departments, sales and other support services in Eurasia and in the USA.

The development department is organized as a set of teams, each responsible for one of the main modules in the product line. A team is mostly a fixed group of 4–6 developers and a team leader, who is experienced in the domain and the technology. The teams share some supportive services such as the Chief Technical Officer (CTO), a system architecture team, and QA-services (quality assurance).

Being a product line organization means that strategic planning is a natural and important activity. This used to be a side activity of some of the supportive roles but has now developed into a full-time prioritized internal service. A Product Strategy Group (PSG), consisting of five product managers, is responsible for developing a product roadmap for each of the main modules, and supports their development.

The processes. The overall SPLE process at CSoft can be described as three interacting main processes, each with a different time horizon (Hanssen and Fægri, 2008). First, the PSG drives a continuous *long-term (1–2 years) strategic process*, creating product roadmaps based on input from nearly all parts of the organization as well as several external sources. These roadmaps are high-level plans, or a vision, for the product line looking one to two years ahead. They typically present business cases, key stakeholders to participate in development, and prioritized product qualities, instead of functional requirements or feature descriptions. The main content of these plans is made visible externally to the organization through various meetings with customers and partners, at conferences, and through other channels. Roadmaps do not describe specific design decisions but rather high-level guidelines, which are elaborated when detailed plans are laid out for the development projects. In some cases, customers or related third parties visit the R&D department to have close meetings directly with one or more of the development teams to elaborate ideas and discuss needs.

The second main process is the *agile development process* Evo (Gilb, 2005), which the R&D department follows to manage the approximately one-year long development projects, leading towards the next main release of the product line (all components are released at the same time). Each component team runs an Evo-project, meaning that development is done in fortnightly iterations, and that each iteration delivers new working software. Each iteration ideally starts with a meeting with an external stakeholder to explain and discuss needs and requirements. At the end of the iteration the team meets with the stakeholder again to get feedback on the outcome (new or improved software) from the iteration. Customers come from all over the world, using a web meeting solution (WebEx^{TM}) to communicate as effectively and closely as possible. This is a radical change compared with the previous waterfall approach where feedback was rare.

The third and last process is *the operational process*, which encompasses the day-to-day operations such as support, training, sales and marketing, and high-level maintenance. Apart from being common functions in a product organization they are also highly valuable sources of input to both the strategic process and the Evo development processes as they represent a wide, diverse, and continuous interface with customers.

The product line consists of five main modules, which together support the core business operation of the customers: a value chain of planning, data collection, analysis, and reporting of results. The composition and use of the modules varies according to customer and case. Some modules can be used in any configuration, while the use of others depends on the situation. The software comes with a set of predefined configurations for the most common usage scenarios.

The product line offers an application programming interface (API), which is implemented as standard web-services. Most modules offer an API, which enables clients to integrate the product with other systems, and which is extensively used by other third-party organizations to offer additional software solutions and/or services. More than 60 such partners now base their business partly or completely on using the CSoft product line as a platform through these APIs.

3.5.2. From creative chaos to an emerging software ecosystem

CSoft was established in 1996 and has grown continuously since then. This development has gone through three phases, and has now entered a fourth. This section presents a summary of these phases of development, and it indicates some important milestones in the development of the organization.

The timeline in Fig. 2 shows the main events (Pettigrew, 1990) in the development of the organization, the approximate increase in staff, and the studies of the organization.

1996–1999: "creative chaos". The company initially grew out of a small business providing manual services to very few clients. A simple homemade software tool grew into a solution that could be sold as a stand-alone software product. The main focus of the company changed, and the development of this product became the main objective. At the start, in 1996, there were only a few employees providing the product to a handful of clients. The process can best be described as ad hoc since the main drivers were almost daily interactions with and feedback from customers. A customer request was literally routed directly to the developers. This start-up phase was a 'creative chaos' – that is, it had nearly no plans and no control, but it was undoubtedly extremely creative and productive. The product grew rapidly, not only in terms of features and functionality, but also in terms of defects and complexity. Work became stressful, with little control, and a lot of overtime.

1999–2003: waterfall. As the number of customers increased the organization formalized the development process, mostly according to the principles of the waterfall model (Royce, 1970). This somewhat disreputable approach to organizing software development emphasizes upfront detailed planning of requirements, design, and development. The development is divided in consecutive phases, where requirements are developed into a design, and the design is developed into a software system, which is tested close to deployment. Prior to this, the R&D department was extended with a QA-manager. This structured approach established a certain level of control, and helped the organization in the continuing development of their product, which grew alongside the customer base. After a few years, several problems arose,

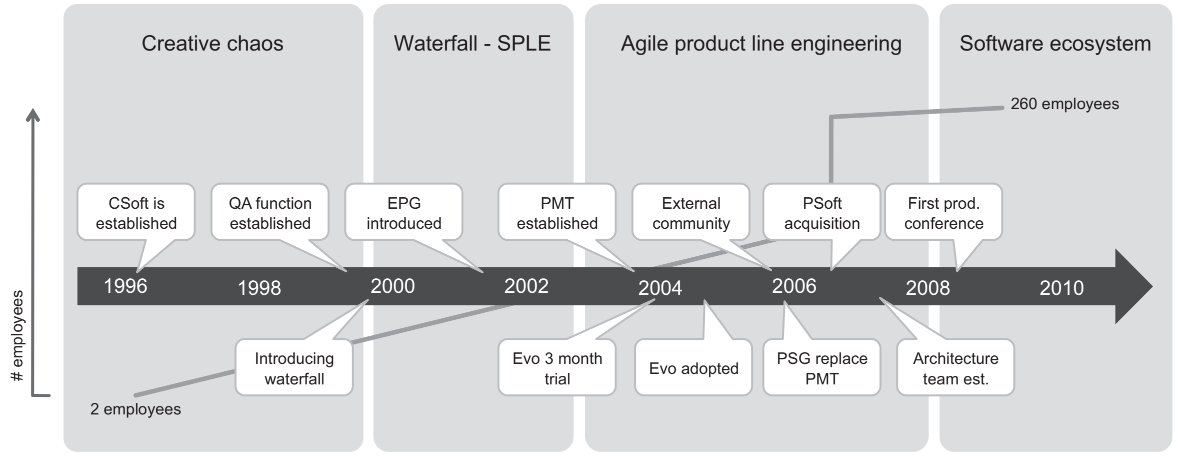

Fig. 2. Timeline of the development of the case organization.

clearly related to the waterfall approach (Johansen, 2005). Testing and verification was postponed to the final stages leading to late identification of problems, which in turn caused much rework. Also, requirements were almost solely focused on functionality, leaving out the quality perspective. To support a growing R&D department, and to spread knowledge about the formalized development process, an electronic process guide (EPG) was developed, and made available via the intranet (Moe et al., 2002). The product expanded and eventually became a product line, capable of serving various usage scenarios. To manage this increasing complexity a Product Management Team (PMT) was formed – a group of experienced employees with other main responsibilities, who were supposed to spend part of their time in strategic product planning. In addition various specialized functions, beyond software development, were introduced such as Technical Account Managers (TAM), the operations department, and training services.

2004 to approx. 2009: agile product line engineering. Due to a critically declining process performance, management of R&D was looking for a way to improve the situation. At a software engineering conference, a few representatives from the company learned about evolutionary development and the Evo method (Gilb, 2005). As it seemed to address some of their concerns they initiated a three-month testing period of this radically different development approach in one of the release projects. Instead of an extensively prepared process adoption, they started out with a few principles, focusing on requirements management, where functional requirements were replaced by explicit expression and evaluation of product qualities, preferably stated by customers involved in the development process (Hanssen and Fægri, 2006). Early experience showed that the number of issues near release was reduced, and that the delivered product matched customer expectations better than before. After this initial process trial, Evo was adopted on a permanent basis (Fægri and Hanssen, 2007; Hanssen and Fægri, 2008). Alongside the growth in the organization and the product line the PMT group was re-established as a full-time Product Strategy Group managed by a Chief Strategy Officer (CSO). Another supportive service, the architecture team, was established, originally with three full time members. Their task was to handle the excessive level of system entropy (Hanssen et al., 2010), and to support R&D in architectural issues. In 2006 CSoft acquired a former competitor (PSoft), and boosted the number of employees to 260. Adding new offices, for both R&D and marketing, was a considerable challenge. Through extensive internal training in the following year, the new organization was mostly using Evo as the development process.

4. Results

This section structures and summarizes the results obtained from the recent study of CSoft. First, we look at how the present product line organization relates to its clients. Then we describe the recent emergence of a community of third party organizations. Together, these results show how CSoft relates to its external environment, constituting a software ecosystem.

4.1. Engaging customers

An important aspect of the continuous change over the past years is how CSoft now relate to their customers. The shift from a plan-driven approach to an agile approach has included an increased proximity to the customers. The initial experience from collaborating with customers as external stakeholders in development projects (Hanssen and Fægri, 2006; Fægri and Hanssen, 2007) showed positive effects such as better management of requirements, and improved motivation among developers. It also showed that the relationships with the stakeholders were fragile, and that it takes continuous and careful management to maintain their motivation to participate. These initial lessons inspired CSoft to further develop and actively exploit close relationships with their customers.

The main motivation for customers to spend time participating in the development projects is the ability to affect development: no payment or any other compensation is provided. One of the product managers explained:

> *"... they see their wishes or their requirements or whatever in the product at the end. And then you get very nice feedback like 'I can see that I said this and that, and in the next release you did it'".*

This collaboration forms a self-regulating system where the supplier and the stakeholders mutually adapt to each other through their shared interest in developing the software product line. This usually works well, but there is always a risk of having external stakeholders, which do not provide the necessary input, as explained by the manager of the PSG team:

> *"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."*

The PSG manager also explained that it is relatively easy to discuss ideas, but that it is more of a challenge when they are included in the development process:

"There's no problem to get them to discuss high-level plans, but it varies when it comes to the development process"

Maintaining the motivation for participating is an important task for the PSG.

Interestingly, large and leading customers tend to expect and demand to be more and more involved at both planning and development stages. One example is a product conference keynote given by the VP from one of the large customers. He stated several 'requirements' (this was the word he used) for being involved, for example:

"Regular meetings with product development teams", "To work as a stakeholder on new software develxopments that are key to us.", "Help to guide product strategy.", and others.

Finding the "right" stakeholders for participation in the development projects is not done through a formal and structured process, but is mostly based on the collective knowledge about the customers. The PSG manager says,

"We don't have a formal process for selecting stakeholders. We have internal discussions, listen to sales people etc. We know which customers have asked for certain features or improvements or those that are heavy users of a particular type of functionality."

In addition, experience from previous participation is also useful as explained by the R&D manager:

"We have become better at selecting [external stakeholders]. Those that have disappointed us are never asked again. You end up with a pool of persons that you know you can trust."

In the very start when Evo and collaboration with external stakeholders were at an experimental stage it took quite some effort to recruit and stakeholders to the development projects and to keep them active (Fægri and Hanssen, 2007). After some releases where collaboration with external stakeholders have become an integrated part of the development process the situation is turned upside-down. When asked to explain this relationship, one of the product managers told us:

"It's almost a problem because as soon as you offer the capability of being a stakeholder, the hardest part is rejecting people, turning them away active participation. So people are very keen on participating."

Co-creating the product line is one of the most significant effects of engaging and communicating with external stakeholders. The rationale is simple, CSoft have the most up to date knowledge of the technology, and the ability to make use of it in the development. Likewise, customers hold the most up to date knowledge of their own business domain, and how it seems to develop. These two pools of knowledge and competence are joined in several ways. One important arena for sharing and gaining knowledge is the product conference where management, strategists, developers and other internal actors get to meet externals from various customers and third parties. Equally important – customers can meet other customers, third parties can meet customers or internals, etc. This shows the networked character of the ecosystem that is shaped around the product line. One example, from the product conference: *several providers of third party products and services were having stands at the conference, communicating with both existing and potential customers and developers from CSoft.*

Another major event, which is more directly focused on the development of the product line, is the annual Advisory Board meeting. Top management from some of the largest and most demanding clients meet with the PSG and other actors who are involved in the shaping of the product strategy. A PSG member explains that they meet to:

". . . discuss high level product strategy and how the demands of their companies and the market are developing." Bringing together major competitors like this was a daring thing to do according to the PSG manager: "The first time we did this it was a bit exciting – would they discuss issues openly, and would they open up? It turned out that they did very fast. They have many concurrent needs, and even though they are competitors they see the value of doing this."

From a practical point of view, we see that tools and infrastructure for collaboration are important enablers for co-creating the product line. Especially the Webex online meeting solution lowers the threshold for having frequent and detailed meetings with stakeholders:

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.

Close corrective feedback in the Evo development projects is another approach to co-creating the product line, but on a tactical level. 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."

This demonstrates one important function of the agile process; the development teams get nearly immediate (within two weeks) and detailed feedback. This closeness to a few selected customers means that CSoft must also consider the needs of other customers, as they are the referent organization, which always has the last word in the development of the product line. This is partly achieved through Evo's focus on product qualities instead of product features, which are typically emphasized in plan-driven development methods. This is a useful abstraction, and it turns the focus from predefined design (features) to effect and impact (qualities). Both the product roadmaps and the evaluation meetings at the end of the Evo iterations evaluate the product qualities. This means that both the development teams and the external stakeholders have to consider *why* something is needed, leaving the *how* to the developers. The PSG manager explains:

". . . we take one step back, and try to think about why our stakeholder needs this, and then rethink other ways of solving their problem. It is in this type of process that the smart things can turn up – that your thinking is totally new and that you come up with a solution which may be a totally different way of doing it, maybe faster . . .".

Catching and following up on customer ideas on an ad hoc basis is equally important as involving customers in regular processes such as roadmapping and the Evo development projects. At the product conference:

A customer representative told about a case where his company gave input to CSoft on some changes they would have liked to see. This led CSoft to invite a delegation from the (abroad) customer to the R&D department in Oslo. Ten CSoft people spent the whole day discussing the solution with four representatives of the customer. This was perceived very positively, and in the end actually affected the software.

Some of the largest customers may also request dedicated workshops to discuss needs and ideas. The PSG manager talks about this:

". . . for some of our largest customers, mostly by their initiative, we organize workshops once a year, usually on a strategic level. They want to know what the roadmaps may bring for the next couple of years, and talk a lot about their needs etc.".

The close contact with customers is also a valuable source of learning about competing solutions. One of the team leaders talks about customers visiting the R&D department:

"In these meetings they demonstrated the solution they used today, and actually demonstrated how they used the competing solutions – what worked well and what needed improvements, as well as ideas they might have. These meetings gave the team a wealth of details, and it was quite clear what to deliver to the stakeholder."

The PSG manager also explains the value of learning from the use of competing solutions:

". . . alternatively they do it using other tools today when not using our solution. The option to work more closely with them and to get that knowledge made us more capable to meet their needs better than before when the development was more of the black-box type".

Learning the business processes and domain is another valuable outcome from the direct contact with selected stakeholders. One of the developers talks about one of the stakeholder meetings in an Evo project:

". . . we have tried to solve a task in a way we believed would be reasonable, but to people who actually use this it is obvious that we have misunderstood the process. This gives us guidance as early as possible."

This illustrates the shared interest that the customers and the supplier have – customers want to learn about the product line and its development. Correspondingly, CSoft learns about the business that their product line is supporting.

4.2. *An emerging third party community*

At present around 60 external organizations base their business completely or partly on CSoft as a platform. This can be value-adding solutions or products, related services, and consulting. Examples are solutions for data visualization or voice data capture technology, assistance in using various components in the product line, and training. This networked community (Fricker, 2009) has not been planned and deliberately established by CSoft, it has emerged spontaneously over the past years. This emergence is mostly driven by customers' need for additional features and services on one hand, and the opportunity to extend and use the product line as a platform on the other hand. Also, building solutions and providing services based on the product line means that external organizations get immediate access to a large group of established users of the product line.

Providers of third-party solutions are considered to be important external stakeholders, and are included in the development of the product line in very much the same way as customers.

"During a product conference, a representative from a third party company, delivering an integrated product, explained that when they needed to improve the integration with the CSoft platform they took on the role of an Evo stakeholder. Communication was mostly done by phone, supported by web meetings with screen sharing."

Offering an efficient integration technology enables a third party community. Over the past few years a set of simple APIs have been offered to enable external actors to make extensions to the product line. The development of these APIs have followed the development of the product line, where each new release has improved existing and offered new APIs due to requests from external actors. This means that there is a long (a year) connection time between a request for an interface and its actual release. As more and more externals have made use of this connection point to the software it has been given increasingly higher priority in the development of the product line. An excerpt from one of the roadmaps exemplifies this:

"We are in dialogue with some clients/prospects who are building their portal in a Content Management System, and need to integrate content from module X [name removed for anonymity] into it. Some competitors seem to have APIs that are easier to use than our SOAP[1] based APIs, making it easier to integrate with other portals/communities. It is therefore an ambition to provide an easier API for including module X content into an external portal."

Due to the extensive use of the APIs by externals and their increasing demand for integration with the product line it became clear that the simple web-service based interface had become obsolete. This has led CSoft to develop and offer a new API called FlexibilityFramework (FF), which enables a closer integration to core services in the product line than the previous (and still existing) simple messaging-based APIs offer. A recent webcast, where the CSO presents FF explains further the motivation for this improved interface:

"CSoft is like a supertanker. It is large, can take huge loads, travel far, and take heavy weather. These are all very positive things, on the other hand, the consequence of that approach is that we are quite careful at looking after the supertanker. That means various procedures, on policy, on quality assurance and so forth. And that means that we get less nimble than we would like. The question we posed ourselves is how can we behave like a speedboat while having all the benefits of the supertanker? I'd like you to think of FF as the speedboat. The tanker is still there. It will still take heavy loads and perform extremely well, but in order to be nimble we can build a few speedboats. And they have independent lives from the supertanker and can run on different development schedules."

The last argument is worth a comment; with this new interface to the product line external actors are disconnected from the long release cycles of the product line, and can develop value-adding solutions independently. This is likely to further drive the growth of the third party community.

Actively supporting the community has become a regular activity in addition to the continuous development of the product line. As this community has emerged and grown, CSoft have seen its value, and started to actively support it. In 2007 a dedicated web-portal was launched to make this community visible and each partner is listed and presented. There are five types of partners, those offering technology that is integrated with the product line, those offering value adding services, some can prepare the use of the product line, some can use it on behalf of clients and some offer consultancy services.

5. Discussion

We have now described the CSoft case, emphasizing the present organizational set-up, its processes and the product line, the development timeline of the organization (Section 3.5.2), and how the present organization relates to its external environment (Section 4). Using this insight we now seek to provide answers to our first research question: *Why and how is software product line engineering developing towards a software ecosystem?*

[1] Simple Object Access Protocol, http://www.w3.org/TR/soap12-part1/.

First of all, the initial motivation for changing the waterfall-like development process by adopting Evo in 2004/2005 was that CSoft struggled with unstable requirements incurring high costs due to little flexibility in the process. Much emphasis was given to extensive and thorough requirements engineering upfront, but with little effect (Hanssen and Fægri, 2006). The immediate experience from involving stakeholders in the short Evo development iterations was that developers felt more comfortable and secure by having this close and continuous dialogue on requirements and results (Hanssen and Fægri, 2006). However, in the first release projects using Evo, it became a considerable challenge to maintain the motivation of the external stakeholders throughout the project. The new process was fragile (Fægri and Hanssen, 2007).

(Change 1) From our study we see that this has clearly changed; now external stakeholders are keen to participate – CSoft actually has to turn down candidates. This change is the result of a learning process that has progressed during the first years of using Evo – customers have gotten to know of this practice and some have gained experience as stakeholders. The engagement of customers and users is generally considered to be an important success factor in any kind of software development (Keil and Carmel, 1995; Chiasson and Green, 2007).

(Change 2) We can also observe another change that took place internally at CSoft. The first experimentation with Evo was only done as an R&D-internal matter, like a kitchen experiment. However, as this turned out to be an improvement of the development practice, this way of working eventually became adopted in the rest of the organization. Now, all parts of the organization, from operational support to the top management, are supporting this practice. An example is the CEO explaining the software development process Evo and its strategic importance in his keynote at a large product conference. Another example is the strengthening of the PSG, which has a liaison function between customers and development teams. This tells us that changing a product line organization takes effort and time, and that both internal and external actors need to learn from practice to accept this opening of the organization and its work processes.

(Change 3) Another change we can see from the results is an increasingly higher external visibility of plans and strategies. Initially this kind of information was kept internal, but it is now more and more openly communicated through various channels. It has turned out that doing this does not introduce the presumed risk of leaking vital information to competitors, but that it is rather an advantage as external actors see what might be coming, they can relate it to their own business, and potentially respond to it.

(Change 4) Another related change is the opening of the product line at the technical level, first with the SOAP-based APIs and now the recent and more efficient Flexibility Framework. Initially this represented a minimal and very limited opportunity for extending the product line, but it quickly grew to a considerable extent as it represented tangible business value. This aspect has eventually been given more attention, and has been designated as strategically important in some of the roadmaps. We see several benefits from allowing externals to use the product line as a platform. Firstly, it increases the variability of the product line – it can be used in more specialized ways, serving more needs. Secondly, existing users represent a great opportunity to the third parties (being the second component of a symbiosis-like relationship). Thirdly, letting externals deal with specialization and minor extensions enables the product line organization itself to maintain focus on developing the core product line. This may be the most important effect (Zook, 2010).

To recap the first research question - this change and the organization it has resulted in explains *why and how* software product line engineering at CSoft has developed towards a software ecosystem.

Why	(1) Customers expect and have learned to value to be involved in development and in product strategy making. (2) A plan-based development approach is unfit when serving a volatile domain where the product line is under continuous and extensive development. (3) The total demands and requirements from customers can become too high for one product line organization to manage alone.
How	(1) CSoft learns about the business it serves through active collaboration with customers and third parties. (2) CSoft makes strategy and plans visible externally. (3) The technical interface of the product line is opened. (4) Both customers and value-adding third parties are considered as external stakeholders. (5) CSoft actively support and assist the community of third parties.

5.1. *Implications for theory*

Software ecosystems, as a concept, have the potential of becoming an important field of practice and research in the years to come. To contribute to the development of this field in general, and to answer our second research question in particular: *What are the characteristics of software ecosystems?* – We propose to shape a theoretical platform to be used in future research. Just like the taxonomy suggested by Bosch (2009), a theory of software ecosystems is valuable and useful to generalize the concept and bring together results from more empirical studies. The theory may over time develop towards a unified and empirically justified understanding of the concept.

Fortunately, the theory of organizational ecology (Trist, 1977), briefly presented in the background section, seems to fit well as a starting point. It concerns organizations operating in complex and unstable domains, in principle a suitable description of software ecosystems – and certainly of CSoft. Using this general theory of *organizational ecology*, we derive a set of theoretical propositions suitable to *software ecosystems*:

1. Member organizations in a software ecosystem are linked to a key organization among them, which acts as *a central referent organization*, doing so even though many of them are only partially under its control or linked to it only through interface relations. (This is the Class 1 system according to Trist's classification.) CSoft is an example of such a referent organization. None of the external organizations are formally controlled by CSoft. However, all activity in the ecosystem is related to the product line, which is controlled by CSoft.
2. Software ecosystems promote *self-regulation*. Our study of CSoft show that the collaborative approach can be seen as a self-regulating system in that the referent organization to a large degree adapts to its external environment, and that the external environment adapts to the referent organization. This is in contrast to the previously centralized control that was applied in the development of the product line.
3. Software ecosystems have a *networked character*. CSoft and its external environment constitute a network of customers and third party organizations. Even competitors may be considered a part of this network, although this aspect has not been studied in particular here.
4. Software ecosystems *exist through the use of technology*. The ecosystem, which CSoft is a part of, relies on the use of technology to enable collaboration. Examples are web-meetings, web-casting, and the software-as-a-service deployment model.
5. Software ecosystems have *shared values*. In the CSoft ecosystem the software (product line) is this shared value. For CSoft, the value is revenue from licenses and services, for the customers the value is improved business operations, and for the third parties the value is revenue from sales of value-adding solutions. This common interest in the shared value creates motivation to collaboratively care for the shared value.

These five propositions constitute a start of a theory for software ecosystems. In addition to these principles adopted from Trist's work (Trist, 1977) we also propose two extensions:

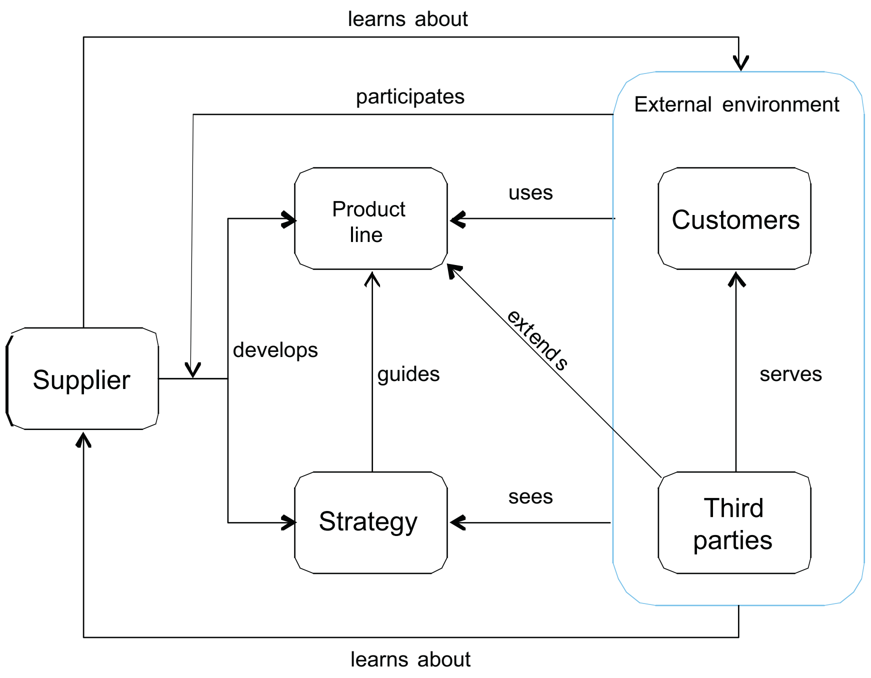

Fig. 3. A conceptual model of a keystone-centric software ecosystem.

6. *The shared value of a software ecosystem is both the software product and the business domain*. Through the increased proximity to the external environment, CSoft have an interest in both the product line *and* the business it serves. Likewise, customers have an interest both in their business *and* the technology they use to drive it.
7. As a software ecosystem emerges, *the control of and influence on its development becomes a shared responsibility between the supplier and the external environment*. An opening of the product line process, with external stakeholders participating in development and with external visibility of plans and strategies, means that some of the control and influence move towards customers and third-parties. This affects the motivation to collaborate, and is of benefit to all members of the ecosystem.

5.2. Implications for practice

From the analysis of our findings we derive a set of implications for practice, relevant to other software product line organizations similar to CSoft:

- Support the external environment by sharing information on plans and strategies – this opens a channel for valuable input and enables collaboration with externals.
- If appropriate, encourage and support a third party community; it can be a valuable extension to the normal development of the product line.
- Establishing and benefiting from a software ecosystem takes time. A successful development relies on repeated cycles of experimentation and learning. This learning process needs to involve all types of actors.

Based on our findings we can derive a strategy that can be of practical value to other product line organizations. The shared interest in the product line (the shared value) is a key enabler for driving the collaboration between the actors in the ecosystem. Thus, a viable strategy would be to (1) make the product line supplier more involved in the development of the business domain and (2) make external actors more engaged in the development of the technology.

To summarize our study we propose a simple conceptual model of a software ecosystem. This model represents the case we have studied, and could serve as a basis to reflect on and guide other similar cases (Fig. 3).

The model illustrates the main actors in a software ecosystem of the type that CSoft is part of – we define this ecosystem as a “keystone-centric” type. The rectangles in the model represent actors and artifacts. The arrows represent activities connecting these roles and artifacts. An activity may also relate to another activity.

Note that there are several other potential variations, with more loosely coupled communities controlling the ecosystem. However, for the CSoft case we see that it is the supplier (key organization) that develops the product line. This development is guided by a strategy, which points out needs and opportunities and main paths of development. Both the strategy and the development of the product line are to some extent visible to external actors. These consist of (at least) customers and third parties, but can also involve others. Third parties use the product line as a platform to serve customers with additional solutions and services. Being a part of an ecosystem means that these actors learn about each other. The supplier learns about requirements, needs, ideas, opportunities, etc. In return, external actors learn about the development of the technology of common interest (the product line), and may even participate actively in the development.

5.3. Limitations

The case study of CSoft is subject to three limitations.

Firstly, this is a single case study, which naturally affects the generalizability of the conclusions. Yet there are good reasons for selecting such an approach. First of all, the number of relevant cases is still low. 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 (2002) discusses the single case study design (pp. 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, at least for the researcher who conducted the study. According to Yin, such a study may act as a prelude to further studies of a relatively new topic, such as software ecosystems in this case.

The second 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 naturally due to the relatively long duration of the study.

The third 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, 2009). Thus, results do not necessarily apply to all other types of software ecosystems.

6. Conclusions and directions for future research

Over a period of approximately five years we have studied a software product line organization and its external environment, showing and explaining an emerging software ecosystem.

We conclude that the development that has occurred has produced effects both internally in the product line organization and in its external environment. The change has led to an increase in collaboration across (previously closed) organizational borders, and it has developed a shared value consisting of two components: the technology (the product line) and the business it supports. Opening up both the technical interface of the product and the organizational interfaces are key enablers of such a change.

Based on our study we propose the following seven directions for future research:

(1) As Jansen et al. also point out (2009a) we need to see more empirical studies of various types of software ecosystems, how they develop, and the effects they produce. Such studies should naturally be focused towards the industry, and be longitudinal as well as exploratory. In particular it would be valuable to get a better understanding of ecosystems as seen from the point of view of external actors.
(2) To build a common understanding of software ecosystems: how they shape, how they work, and what their effects are, we advise a further refining of a theory of software ecosystems, as the one proposed in Section 5.1. One way of developing these concepts would be to apply them to existing well known ecosystems such as iPhone app store, MS CRM, Eclipse, Android and others.
(3) The emergence of software ecosystems comes with new business models affecting intellectual property rights, economic models, competition, etc. We need to see more dedicated studies of these issues to realize the potential of ecosystems.
(4) Software ecosystems are closely related to the more mature concept of open source software development. We need to better understand the similarities and the differences in order to transfer knowledge between these two related domains (Fitzgerald, 2006).
(5) The engine of a software ecosystem is the collaboration with external actors. We have showed some examples through our studies, but this is a broad topic that needs further investigation.
(6) The study of software ecosystems potentially relates to several disciplines such as business strategy, sociology, technology and innovation management, economy, and others. We have briefly touched a few of these and we see a need to investigate these links further.
(7) Software ecosystems affect the shape of control structures. We believe that control shifts from the supplier towards the users, a transition that needs to be better understood.

Acknowledgements

We are grateful to the people at CSoft for sharing their precious time, and providing highly valuable input to this study. This work was done as a part of the EVISOFT project (grant 156701/220) and the Agile project (grant 179851/I40), both partly funded by the Research Council of Norway. We also greatly appreciate the feedback from the anonymous reviewers, which have helped in improving this manuscript.

References

Alspaugh, T.A., Hazeline, U.A., et al., 2009. The role of software licenses in open architecture ecosystems. In: Jansen, S., Brinkkemper, S., Finkelstein, A., Bosch, J. (Eds.), First International Workshop on Software Ecosystems (IWSECO). CEUR-WS, Falls Church, USA, pp. 4–18.
Bosch, J., 2009. From software product lines to software ecosystems. In: 13th International Software Product Line Conference (SPLC'09),. IEEE Computer Society, San Fransisco, USA, pp. 111–119.
Chesbrough, H., 2006. Open innovation: a new paradigm for understanding industrial innovation. In: Chesbrough, H., Vanhaverbeke, W., West, J. (Eds.), Open Innovation: Researching a New Paradigm. Oxford University Press, Oxford, pp. 1–12.
Chiasson, M.W., Green, L.W., 2007. Questioning the IT artefact: user practices that can, could, and cannot be supported in packaged-software designs. European Journal of Information Systems 16, 542–554.
Emery, F.E., Trist, E.L., 1965. The causal texture of organizational environments. Human Relations 18, 21–32.
Enkel, E., Gassman, O., et al., 2009. Open R&D and open innovation: exploring the phenomenon. R&D Management 39 (4), 311–316.
Fitzgerald, B., 2006. The transformation of open source software. MIS Quarterly 30 (3), 587–598.
Fricker, S., 2009. Specification and analysis of requirements negotiation strategy in software ecosystems. In: Jansen, S., Brinkkemper, S., Finkelstein, A., Bosch, J. (Eds.), First International Workshop on Software Ecosystems. CEUR-WS, Milan, Italy, pp. 19–33.
Fægri, T.E., Hanssen, G.K., 2007. Collaboration and process fragility in evolutionarily product development. IEEE Software 24 (3), 96–104.
Gilb, T., 2005. Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage. Elsevier Butterworth-Heinemann, Burlington.
Glaser, B.G., Strauss, A.L., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction, New York.
Hanssen, G.K., Fægri, T.E., 2006. Agile customer engagement: a longitudinal qualitative case study. In: 5th International Symposium on Empirical Software Engineering (ISESE'06),. ACM, Rio de Janeiro, Brazil, pp. 164–173.
Hanssen, G.K., Fægri, T.E., 2008. Process fusion – agile product line engineering: an industrial case study. Journal of Systems and Software 81, 843–854.
Hanssen, G.K., Yamashita, A.F., et al., 2010. Software entropy in agile product evolution. In: 43d Hawaiian International Conference on System Sciences (HICSS'10),. IEEE Computer Society, Hawaii, USA, pp. 1–10.
Jansen, S., Brinkkemper, S., et al., 2009a. Business network management as a survival strategy: a tale of two software ecosystems. In: First International Workshop on Software Ecosystems (IWSECO),. CEUR-WS, Falls Church, USA, pp. 34–48.
Jansen, S., Brinkkemper, S., et al., 2009b. Introduction to the proceedings of the first workshop on software ecosystems. In: First International Workshop on Software Ecosystems,. CEUR-WS.
Jansen, S., Finkelstein, A., et al., 2009c. A sense of community: a research agenda for software ecosystems. In: 31st International Conference on Software Engineering (ICSE'09),. IEEE Computer Society, Vancouver, Canada, pp. 187–190.
Johansen, T., 2005. Using evolutionary project management (Evo) to create faster, more userfriendly and more productive software. Experience report from FIRM AS. In: Bomarius, F., Komi-Sirviö, S. (Eds.), 6th International Conference on Product Focused Software Process Improvement (PROFES'05). Springer Verlag, Oulu, Finland, pp. 216–223.
Keil, M., Carmel, E., 1995. Customer–developer links in software development. Communications of the ACM 38 (5), 33–44.
Klein, H.K., Myers, M.D., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Quarterly 23 (1), 67–93.
Messerschmitt, D.G., Szyperski, C., 2003. Software Ecosystems, Understanding an Indespensable Technology and Industry. The MIT Press, Cambridge.
Moe, N.B., Dingsøyr, T., et al., 2002. Process guides as software process improvement in a small company. In: EuroSPI, Nuremberg, Germany, pp. 177–188.
Pettigrew, A.M., 1990. Longitudinal field research on change: theory and practice. Organization Science 1 (3), 267–292.
Qualman, E., 2009. Socialnomics: How Social Media Transforms the Way We Live and Do Business. John Wiley & Sons.

Royce, W.W., 1970. Managing the development of large software systems. In: IEEE WESCON, Loas Angeles, USA, pp. 1–9.
Seaman, C.B., 1999. Qualitative methods in empirical studies in software engineering. IEEE Transactions on Software Engineering 25 (4), 557–572.
Trist, E.L., 1977. A concept of organizational ecology. Australian Journal of Management 2 (2), 161–175.
Trist, E.L., Bamforth, K.W., 1951. Some social and psychological consequences of the longwall method of coal-getting. Human Relations 4 (1), 3–38.
Yin, R., 2002. Case Study Research. Sage Publications Inc., Thousand Oaks.
Zook, C., 2010. Profit from the Core: A Return to Growth in Turbulent Times. Bain & Company, Inc., Boston.

Geir Kjetil Hanssen works as a senior research scientist at SINTEF ICT, Norway's largest independent research institution. He holds a PhD degree in informatics from the University of Trondheim (NTNU).