Chapter to appear in "Activity Theory in Practice: Promoting Learning across Boundaries," in H. Daniels, A. Edwards, Y. Engeström, T. Gallagher, S. Ludvigsen (eds), Taylor & Francis Books, UK.

Adaptation and Generalisation in Software Product Development

Anders I. Mørch, Kathrine A. Nygård, Sten R. Ludvigsen
InterMedia, University of Oslo, Norway

Summary: This chapter presents a case study in software product development in a company that develops and sells project-planning tools for the oil and gas industry and provides consultancy services, training and support in using these tools. We were invited into the company to give advice on their knowledge management practices for customer relations. The rationale for the invitation and the subsequent intervention is twofold. On one hand, the company is in transition to expand to new markets in order to increase revenue. On the other, they want to maintain good relations with existing customers. We analyse these activities, which we have labeled "generalisation" and "adaptation," respectively. Our empirical material consists of interview data and a video recorded meeting with key stakeholders. The conceptual framework for analysis combines Activity Theory and Evolving Artifacts. We illustrate our findings by the development of two software products (Planner, Microsoft Project Extension). Generalisation occurs at uneven intervals and represents major changes for the company, whereas adaptation occurs more frequently, is incremental in terms of change, and is often initiated by customers.

INTRODUCTION

This chapter analyses data from a case study in software product development in a software house in Norway gathered over a two-year period. The software house is referred to as the Company. The company is engaged in commercial software development and develops and sells project planning and management tools and provides consultancy services, training and support in using these tools. At present, the company employs 25-30 people, but it is rapidly expanding and searching out new market share. The main market has been the Nordic oil and gas industry. To expand to new markets, in particular building and construction, the company has started to change and improve its knowledge management practices regarding customer relations, which have rested largely on oral and personal connections. These practices are time consuming and not aligned with the goal of serving a growing market with diverse customers.

The first attempt to improve knowledge management practices involved implementing an automated Helpdesk, but it did not work well (little used). The goal of this technology was to allow customers to send email requests when they needed help with one of the products instead of contacting one of the consultants or developers by telephone. Although the interface to the automated Helpdesk was very simple, the customers found it more convenient to contact the persons they knew from when they purchased and installed the project management tool.

The second attempt was an interactive Web or Web 2.0 (web portal for short) to be integrated with a new Customer Relations Management (CRM) tool that the company intended as a combined communication hub between the two national offices, a link between consultants in the field, and to support customer interaction (Nygård & Mørch

2007). We entered the company in conjunction with this initiative and built a prototype of the web portal on an open source application framework that was tested in the company by its employees (Nedic & Olsen 2007). The project stranded when we were not able to integrate the portal with the CRM interface (a vendor product that was not open for integration with third party tools without extensive debugging and work around).

The partial failure of these two efforts of supporting knowledge management of customer relations calls for a deeper analysis, which goes beyond the analysis of the use of software tools. It has become clear to us that the tensions between the different developmental practices could not be solved by one type of technology. The problem calls for technical solutions at different levels in the organisation, and for integrating the solutions with organisation design.

In our view, the problem stems from two interdependent activities that are only partially coordinated by the company. One is customer-initiated product development, which occurs frequently as many small short-term efforts that contribute to incremental development of the products. The other is software product line development, which happens less frequently but lasts longer, and represents major changes for the company (such as new products added to the existing line of products). Our working hypothesis is that these are both prioritised software developmental activities but they are not compatible with each other because they operate on different time scales and employ different strategies: one *adaptive*, the other *expansive*. This incompatibility is a research problem that needs to be unraveled before new knowledge management practices can be successfully implemented in the company.

Our research questions are formulated as follows:

- How does the company expand its business to new markets and its products to new users?
- How does the company interact with customers to leverage their contributions to development?
- How can we understand these two activities in terms of how their components interact (e.g., to what extent are there paths, stages, transitions, breakdowns, contradictions, deliberation, resolution, etc.)?

RELATED WORK

Eric von Hippel has studied users as innovators and their impact on product development, referred to it as user-driven innovation. He has developed a method for following “lead users” (von Hippel 2005). A lead user is someone who is modifying a product to adapt it to a new situation based on knowledge of a related product or practice, or an early adopter and champion of a new innovation. Von Hippel has found that lead users’ innovations often become commercial products (von Hippel 2005). For example the motocross series of bikes manufactured for teenagers during the late 1960s and early 1970s originated as the result of teenagers’ desire for their bikes to resemble adult motocross bikes. In the area of software development, participatory design (Ehn & Kyng 1991), directed observation (Norman 2008) and strategic ethnography (Pollock &

Williams 2008) are methods for addressing similar issues. Directed observation means to seek out and analyse the workarounds, hacks, and clever improvisations lead users and ordinary people create at work and at home (Norman 2008). Strategic ethnography is longitudinal studies following artifacts (packaged software) as they evolve over time, which has been referred to as capturing the biography of these artifacts (Pollock & Williams 2008).

The SER (Seeding, Evolutionary growth, Reseeding) model is a process model for integrating end-user development with software engineering. It is different from user-centred design (e.g. prototyping) and from software engineering (e.g. specification-driven methods) (Fischer & Ostwald 2001). The goal of integrating these two types of software development activities according to Fischer is “meta-design,” providing end users with tools that allow them to tailor and further develop domain-specific tools (Fischer 2007). These “meta tools” or design environments as Fischer calls them are created by professorial developers and integrated with the domain-specific tools. These tools can provide a new dimension of human-computer interaction by allowing users to interact with applications at different levels of abstraction (from use to development). This requires a shift in mindset when it comes to the roles of developers and users, requiring new roles that expand and transcend established boundaries (e.g. end-user developer, super-user, developer working in a user context).

Framing our own conceptual framework and perspective are the studies of “super users” we previously conducted in two companies. In one of these cases Åsand & Mørch

(2006) followed the activities of super users and local developers during the adoption of a new business application by an accounting firm. This company launched a program to train super users so that could master the complexity of the new technology, a generic, multi-purpose application replacing several older, non-integrated standalone applications. End-user development activities were documented empirically and analytically, using interviews to gather data and drawing on aspects of Activity Theory for the conceptual framework for analysis. The findings indicated a need for closer interaction between end-user development and professional development, because certain instances of end-user development lead to technical improvements that was useful to all of the company's business applications.

In sum, based on related work and previous research in this area, we argue that to understand the two types of processes that we call adaptation and generalisation we need both historical analysis and data that provide us with sufficient detailed information about how these processes are initiated, evolve independently, and are interrelated.

CONCEPTUAL FRAMEWORK

Activity Theory and the Evolving Artifacts approach provide a set of analytic concepts that we have found relevant for our analysis. They are presented and partly developed in this section and are further used in the analysis.

Cultural Historical Activity Theory And Multi-level Analysis

Cultural Historical Activity Theory (CHAT) combines different levels of analysis in studies of how institutional practices change. Historically, activity theory has its origins in the Russian School of Cultural Psychology and the work of Vygotsky, Leontiev and Luria (Roth & Lee 2007). Although there are several contributors within activity theory, the analysis we develop draws on the work of Yrjö Engestöm and his research group and their studies of change in various institutions: companies, hospitals, universities, and schools. Engeström developed his theoretical approach to guide a multi-level analysis of change amid stability in an activity system or institution (Engeström 1987, 1999). The connections between the core concepts in his approach are represented in Figure 1. The central issue with regard to the graphical representation is that it brings forward interdependencies between the acting subject and different levels in an activity system. In other words, it is a framework for analysing a multitude of relations.

FIGURE 1 NEAR HERE

Change within activity theory is connected to the concept of object. The notion of objects refers to the focus of the activity and what it is that gives the activity direction and outcomes. Since every activity is object-oriented it mediates our relation to reality understood as multilevel and heterogeneous and helps to distinguish one activity from another. It is interesting to note that the analytic potential of this concept has been used quite differently among activity theory researchers. In Engeström's version of CHAT (Engeström 1987) the focus is on collective phenomena; he uses the notion of object to scrutinise the dialectical process between the subjects and their communities. This implies that the object is tied to the collective level, works over longer time spans, and signifies the potential and direction for change. In other words it is not static and not reducible to short-term goals but should rather be understood as jointly constructed and potentially, but not necessarily, shared. This means that objects are never fixed and that new and surprising aspects may emerge from the object. The notion of object then introduces a dynamic interdependency between elements of the change process. Objects are tension laden since they are composed of both material and ideal or symbolic aspects, which means that we can only partially describe and understand how they work. Hence, the analytical focus on the transformation of objects brings together the different levels within an activity system and interacting activity systems.

Engeström's account of change is connected to expansive learning, which involves new activities and the transformation of activity systems. Expansive learning includes change in all the elements in the activity system, which means that the subjects develop new types of agency, their use of tools changes, and new rules apply, while the division of labour and how communities operate is also transformed. This shows how the activity system includes individual, interactional, and collective dimensions. In sum, a multi-level approach. This understanding of change together with the dual nature of objects is deeply rooted in the dialectical idea. This implies that systemic historical contradiction is the mechanism that transforms society and its institutions over time. Contradiction is manifest in incompatible relations between components of the activity systems and between activity systems.

Two notions that are related to CHAT and of particular relevance to our study are boundary crossing (Tuomi-Gröhn, Engeström & Young 2003) and co-configuration (Victor & Boynton 1998). Boundary crossing in customer-initiated products development means there are borders between the software company and its customer organizations that are not clear-cut from the point of view of developmental practices (Ludvigsen, Havnes & Lahn 2003; Nygård & Mørch 2007). Instead there is movement between what we later will define as different activity systems. Co-configuration is another concept that is useful in our case. It was first introduced in Victor and Boynton (1998) in order to describe and understand how companies work with customers to deliver products that are tailored for their needs. They argue:

> Doing mass customisation requires designing for a product at least for each customer. This design process requires the company to sense and respond to the individual customer's needs. But co-configuration work takes this relationship up one level–it brings the value of an intelligent and "adapting" product. The company continues to work with the customer-product pair to make the product more responsive to each user. In this way the customisation becomes continuous.
>
> (Victor & Boynton 1998: 195, reproduced in Engeström et al. 1999)

One example of co-configuration work is the development of software applications with end users. Engeström et al. (1999) connects the concept of knotworking with co-configuration. Knotworking signifies an activity that captures a pulsating movement, in which different aspects or threads are connected that would not have been connected otherwise. During knotworking the actors assemble often in an improvised, collaborative activity, without clear institutional boundaries. We do not use co-configuration in the same way as defined by Victor and Boynton, as an intelligent and adaptive product. Instead we use it to explain *adaptation*, how it is the users in collaboration with developers who adapt the technology, not the technology by itself.

Together with the other analytic concepts we think that CHAT provides a solid foundation for analysing how a software company and its products co-evolve, when focussing on the former and the relationship to the latter.

Evolving Artifacts Approach

When tensions and breakdowns occur during development, the company will need to find solutions to work around them. This may require boundary crossing and movement between two activity systems (developer and customer) with the effect that a specific solution at a customer site may become part of the historically given (general) solution developed by the company. We introduce the terms aggregation and emergence to capture the mechanics of this transition. A specific solution is *aggregated* when it is integrated into a higher order (general) solution to resolve the tensions and breakdowns. When this is accomplished (fully or partially), we say the specific solution *emerges* as a "trace" in the product's historical development.

The notion of "evolving artifacts" is a term used in software engineering and end-user development, but originated with the work of Herbert Simon (Simon 1996). It is further developed here for the purpose of integration with activity theory. Simon (1996) made a distinction between normative and descriptive representations in relation to artifacts by saying that “artificial things are often discussed, particularly when they are being designed, in terms of imperatives as well as descriptives.” We use this distinction in the analysis of the informants’ talk about the company’s products. The notion of stable intermediate form (Simon 1996) denotes a building block in the development of evolving artifacts. These building blocks are organised in part/whole (aggregation) hierarchies, and provide a structure for integrating (parts) into higher order solutions.

The evolving artifacts approach has been further developed and applied to software development by Gerhard Fischer and colleagues. According to Fischer and Ostwald (2001), software development is an iterative process of (1) creating, (2) discussing, (3) accumulating, and (4) formalising representations for mutual understanding. Mørch (2003) applies this to evolution of standard (packaged) software applicaions and defines creation by end-user tailoring (Mørch 1996). End-user tailoring gradually modifies existing applications over time without disrupting their inner workings (adding rather than modifying code). Discussion in this context means to decide whether or not a local (tailored) solution should be approved or rejected for inclusion, whereas accumulation and formalising are a developer activity of incorporating locally developed solutions into a higher order solution (product).

This process requires boundary crossing between developer and user organisations, mediated by two types of boundary objects (Star & Griesemer 1989): design rationale and predecessor artifacts. Design rationale is documentation (a description) of software functionality to support developers and end-user developers with the reasoning behind a design to ease reuse and further development (Fischer et al. 1996). A predecessor artifact is a specific solution that has been replaced as a result of improvement and/or innovation (e.g. when a new release of software improves certain features, the current version of these features is no longer needed for running the application). Following predecessor artifacts is a useful method for researching evolving artifacts because they play a central role as boundary objects during evolutionary development, bridging between past and present solutions. If predecessor artifacts are accessible upon demand, they can supplement a new solution in one of several situations: 1) back-up technology during breakdown (Gasser 1986), 2) reminder of a previous way of using the tool (Ehn 1988), and 3) stable intermediate form for further development (Mørch 2003). An example is the coin-operated public phone that became "extinct" in many parts of the world as a result of the spread of mobile phones.

We draw on George Herbert Mead and his notion of emergence to theorise the role of predecessor artifacts for evolving artifacts. He defined emergence as "the presence of things in two or more different systems, in such a fashion that its presence in a later system changes its character in the earlier system or systems to which it belongs" (Mead 1932: 69). Mead used the term to discuss how an individual in a group through interaction with others contributes to the creation of a "generalised other," emphasizing that the total can be more than the sum of parts, with the implication that emergence is about reorganisation, which brings in something extra (Mead, 1934: 198, Dodds &

Valsiner 1997). In our case, the “extra” is the presence of historical information in evolving artifacts (i.e. traces of the past). Aggregation (Simon 1996) and emergence (Mead 1932, 1934) together capture two central aspects of evolving software artifacts as they pass through stages from creation to formalisation during boundary crossing (Fischer & Ostwald 2001).

Our main use of these concepts is to distinguish two levels of software development, general and specific, using aggregation to bridge from specific to general and predecessor artifacts to bridge from general to specific.

METHOD

We have used interviews, observation, and video-recorded workshops and analysed software artifacts and documents pertaining to software development activities. The eight extracts we show below are data from one workshop, two interviews with company respondents and one interview with a customer. The total body of data is 10 ½ hours of video material from 11 informants.

The data is approached as dialogues in order to grasp the meaning creation that is constructed in the situations. The informants draw on their experience from everyday work practices and provide accounts for activities history (Scott & Lyman 1968) by presenting narratives of the relationship between product development and the company’s. These accounts capture tensions and breakdowns in the historical development of the company and give a retrospective description of the actual situations that they refer to. The accounts include clarification, explanation, description of context occasions, elaboration, and justification in specific situations, and we reference this in the analysis to justify our claims. Descriptions of different aspects of the context are used to provide evidence and link together what is treated as historical events and significant changes in practice. This type of talk is generated by the participants in order to be understood, and thus follow the roles and norms of participation in interviews and arranged workshops.

EMPIRICAL ANALYSIS

Levels of Development and Software Product Line

The company is known for its adaptive product development philosophy, i.e., close interaction with customers to develop tailor-made products. The data indicate two interrelated levels of development, general development and specific development. One is associated with professional development and the other end-user development.

General development is the software engineering work leading to new versions of a product, new product titles, and improved practices in software development. We focus on the identification and description of the events that are of historical significance for the company. General development is also about generalisation. By this we mean the engineering and distribution work to "spread" a product from one instance to many, and making it applicable to new application domains as markets come and go. General development is activity at a higher level of abstraction than specific development.

Activity theory is useful at this level because generalisation implies change in direction of the activity system, which means a new historical object.

Specific development is incremental changes made to concrete software artifacts (instances) as a result of adaptation of a product to a specific user organisation as well as the joint developer-customer work to accomplish the changes. Specific development occurs more frequently than general development and on a different time scale. Incremental changes will often be accomplished by a few days' intensive work, often initiated and sometimes accomplished by customers. Tools for bug reporting, improvement requests, artifact annotation, and end-user development support this work. Specific development influences general development and vice versa. Activity theory is useful at this level in order to study boundary crossing of two different activity systems (developer and customer). The details of the boundary crossing have been analyzed through the lens of evolving artifacts.

Understanding the transitions between specific and general development has been central to our research. The informants accounted for it in terms of historical events connected to organisations the company does business with. Some of these events led to major changes in the company, including the creation of two new products, whereas most were minor, albeit ensuring gradual improvement and continuation of existing products. In software engineering and products manufacturing the technical work to accomplish this is referred to as *product line development*. Figure 2 shows a schematic overview of the factors influencing product line development in the company. The picture is referred to and elaborated in the analysis below.

FIGURE 2 NEAR HERE

Introduction To Analysis

The following data extracts are selected on the basis of the two empirical categories generalisation and adaptation. By *Generalisation* is understood how a software product reaches new application domains and is integrated with another product originally developed by a competitor. *Adaptation* is understood as development for specific customers by involving them in various active roles. The former represents major development and historical contradictions and the latter customer-initiated product development. In the following we will provide examples on generalisation and adaptation with eight extracts from the data material.

The main finding indicates a strong link between the unfolding history of the company and the evolution of its products. Furthermore, we can trace evolution on two levels, general (in company) and specific (with customers).

Generalisation

Major Change in Software Product Line

Early on in the project, the data indicated two distinct in-house cultures represented by two regional offices. One (main office) focused on development and sales and the other support, consultation, and sales. We had good access to the office in our region, but the need to visit the main office became evident. A two-day workshop was set up in December 2006, where four representatives of the research group participated in a workshop and held interviews with four key representatives from the main office. The following two extracts elaborate Figure 2, in particular the two main events around year 2001.

Extract 1: The workshop participants discuss collaboration with a big competitor that resulted in a new product, Microsoft Project Extension (MPX, our acronym). The researcher initiates the discussion by asking about the competition the company met when entering a new market.

22. Respondent1: A large competitor in the building industry was Microsoft, but we are partners now, because we built an extension [plug-in, component] to Microsoft Project that has given us access to their market through the product, and we are not really competitors anymore, we are kind of partners.

23. Researcher: Was that the reason you chose to make the extension?

24. Respondent1: It wasn't really so because we were contacted by them.

In line 22 Respondent1 mentions a major player in the building industry for project planning tools, and refers to it as a former competitor. He clarifies why they are no longer competitors by explaining that the company has built an extension to the competitor's product. He points out the benefits the company has enjoyed in terms of market share as a result of this and elaborates by saying that these came through the product. The respondent describes the role of the company as "*kind of partners*" as opposed to being competitors. They are partners in light of providing Microsoft with a technological solution it needs, and this partnership was a result of Microsoft's reaction to losing a major contract in competition with the company. The researcher then asks the respondent if increased market share was the reason they built the extension, which is answered that they were contacted by Microsoft to do so, which is elaborated in the next extract.

Extract 2: The next extract is from conversation with a leader in one of the regional offices. The respondent gives a narrative of the company history, using PowerPoint slides for illustration.

The respondent gives an account of how a particular event laid the foundation for collaboration with Microsoft after outbidding them for the contract.

> The reason we came into contact with Microsoft was that in 2001 Statoil decided to obtain a project management tool for all its PCs, which are a few thousand; it was something like six to eight thousand PCs, and we won the bid with an adaptation of Planner. Microsoft was the one that lost the bid.

The respondent gives a narrative of the company's history in 2001. He gives an explanation of the events prior to the partnership with Microsoft. The company won a bid for a contract with an oil company with an adaptation of the *Planner* project management tool. The prior work with Planner and Project are suggested as the reasons for outperforming Microsoft Project on this bid, and why Microsoft subsequently contacted the company to develop MPX.

The collaboration with Microsoft represents a major expansion for the company, both in terms of a new branch in its product line (MPX was launched in 2001) and for reaching out to new customers (building industry). One of the effects of this was the establishment of an office in the United States. Many of the new customers were familiar with Microsoft Project and/or Project and Planner and appreciated the integration of the two products into one. We elaborate on the expansion to the building industry below.

Emerging New Object

Extract 3: In the next workshop extract, the participants talk about how the company extended their enterprise to the building industry as a result of project managers migrating from one industry to another to discover that the tools they were used to were not widely available in the new domain. The researcher prompts the respondent to expand on the key events in the company's history regarding this transition.

6. Researcher:	You didn’t launch a strategic design initiative, you didn’t adapt the products or make a new product, and these are the same products?
7. Respondent2:	Exactly the same products that are used in another market, but because some of those who have come [into the new market] have higher standards for project management and wanted to use tools they were used to. For instance in Ahus [large hospital being built in Norway] where you have people who have managed projects in the oil and gas industry for 25-30 years; they wish to do things in a certain way, and have also instructed their sub-contractors to report in a certain way, and they [sub-contractors] have to find suitable tools as well. It is the same way project management spread to the oil and gas industry some thirty years ago.

Based on a question by the researcher in Line 6 Respondent2 says that the product is the same when used in the two different application domains (oil and gas and building). This can mean that it is not the company’s developers themselves that initiate change to products, but specific external events in customer organisations.

Respondent2 justifies the introduction of new management tools in the building industry by identifying a need for running construction projects in this area in the same way as in oil and gas. This created a business opportunity for the company because of the relationship already existing between the company and these project managers. He also indicates that project management tools in oil and gas set a standard for project management that transferred to the building industry.

He gives an example from the project of building a large hospital in Norway (Ahus). The company's project management tools could be used without any changes, emphasising the expansion possibility emerging from interdependency among building industry sub-contractors. That this is a rare event is further supported by the last sentence in the extract that says that a similar expansion happened when the founders of the company entered the oil and gas industry in the mid-1970s. In the next extract another plausible explanation will be presented, focusing on the "why" of the expansion.

Extract 4: Later in the same conversation as illustrated in the previous extract the participants elaborate on the expansion to the building industry by providing an account for *why* the shift took place.

8. Researcher: But is it a coincidence that these people from oil and gas who run projects are in the building industry now, or did the building industry realise that the projects were getting more complex, or how else did it happen?

9. Respondent2: I think that one reason was downsizing in the offshore [industry] five to six years ago. Aker [big national offshore engineering

company], for instance, closed down everything that belonged to Aker Engineering in Oslo. And a lot of those people you will now find in the building industry around Oslo.

10. Respondent1: In large projects in the building industry.

The researcher asks for clarification about why project planners migrate from oil and gas to building. He elaborates the question by offering two alternative explanations. Was it a coincidence or was it a deliberate action from the building industry facing increasing complexity in project management?

Respondent2 starts by listing the reasons for the migration by describing a historical incident of downsizing staff in one particular engineering company in the oil and gas industry. He explains how these professionals moved into the building industry in the greater Oslo area as a result of this.

Respondent1 elaborates on this by pointing out that they now have many new customers (large projects) in the building industry as a result of the migration.

In sum, the analysis at this level shows how the company has transformed itself historically in relation to Microsoft and in relation to their changing markets (a moving target). The effects of this for the company are twofold: Establishing collaborative interaction with a previous competitor in order to launch a new product and to get access to a new market, and following the customers as they move from one industry to another. These are long-term cycles of activities based on migration of people and inter-organisational interaction. These activities can be traced to tensions and breakdowns in short-term activities. The tensions grew over time and created dilemmas and contradictions that were solved through new products and the transformation of the company into a new business area. These organisational solutions have effects on both internal and external relations as well as product portfolio. It is also worth noticing that change to the products does not occur frequently, if at all, on this general level of development.

Adaptation

Adaptation is short-term activity and takes place more frequently. It is initiated by customer improvement requests and end-use tailoring of the company's products. Customers submit improvement requests to promote an idea for a new feature to a product or to report a breakdown in use of one of the products, whereas end-user tailoring is a local solution created by users by customisation or domain-specific programming.

Improvement Requests and End-user Tailoring

Extract 5: This extract illustrates a developer's view of user participation in adaptation. It is from an interview with the head of support in the company, who is also involved in sales, development and management. In this extract the interviewer focuses her questions around how user participation is initiated in practice and the respondent refers to one of the products to exemplify user participation.

9. Interviewer1: Are there any customers that have participated in the development of your products?

10. Respondent2: […]Statoil [oil company in Figure 2] is an example. When we delivered version 3 of Planner, Statoil was a major initiator of the development. Much of what we incorporate in our products comes from our customers.

11. Interviewer1: How do you receive customer requests, how is the process accomplished?

12. Respondent2: Customers send us a wish list for new functionality or modifications to existing functionality. During development, if [the request for new functionality] seems reasonable, e.g. if others are asking for it, if it is an area we should look closer into and maybe look at in a broader perspective. For example, if you are writing reports and [someone] wants new functionality, we include it because we are already are in there [altering the report-module in the product]. This enables late requests to be taken into account, assuming it doesn't have side effects, requiring changes to many of the other modules in the product.

The interviewer asks the respondent for a direct account of whether or not customers play a role in development activity. Respondent2 explains how one of their main customers is an active contributor to new ideas for development, and he exemplifies this by referring to one specific version of the project management tool *Planner* (see Figure 2). He later generalises this *("Much of what we incorporate in our products comes from our customers").* The interviewer goes on to ask how improvement requests are received. The critical factors dealing with request processing include whether or not the request is judged to be important for the task at hand, how much extra work is required to incorporate it, and to what extent it is restricted to a well-defined area in the software code. This indicates two levels of development (customer and company), each with its own time scale and change rhythms. The discrepancy among the levels is best reconciled when specific requests for change align with the company's internal development cycles.

There is also an implicit classification scheme of improvement requests that helps to filter improvement requests and channel un-coordinated improvement requests. When received by the company it will first be classified as *good, possible* or *bad* (Andersen & Mørch 2009). A good suggestion is accepted as is. A possible suggestion must be accompanied with payment, and a bad suggestion is rejected outright.

Extract 6: In the next extract it is shown how some customers are able to make changes to the products on their own. It is an example of end-user tailoring. The conversation is about how the *Planner* tool is used in this setting, and the relation the customer has with the company.

1. Interviewer1: Could you tell me more about making your own adaptations; you mentioned there were no improvement requests originating from your company?
2. Customer2: No, we have not contributed to any major changes to the products. A reason for this is that I know SQL [database query language], allowing me to work around the problem of organising tables. Information is available through documentation, allowing me to work around and find access to make the necessary adaptations. I have not modified the functionality as such, but I have been able to solve the problem I need through this workaround [writing in SQL].
3. Interviewer1: Ok.
4. Interviewer2: So, one could say that you did your own local adaptation?
5. Customer2: Yes that is one way of looking at it.

The interviewer asks the respondent why there have been no improvement requests from his organisation as a follow-up to a previous question on the topic. He confirms that no major changes to any of the products has been initiated by his organisation, and clarifies this by referring to his knowledge of SQL (a specialised, user-oriented database query language). This enables him as expert project manager to make local changes and work around an inefficient technical solution. As an example he points to organisation of database tables suitable for his project management work in a way that meets his organisation’s and sub-contractors’ needs. The advantage is that he will be able to see the results of his request for new functionality relatively quickly compared to submitting an improvement request and waiting for response. A disadvantage is that his homemade solution might not run with future releases of the product.

Spread of User Innovation

The next two extracts illustrate how the company takes advantage of the local initiatives to improve the products. The extracts illustrate that there is a connection between adaptation and generalisation, according to how generally useful specific solutions are distributed: *local spread* and *global spread*. Furthermore, the following software engineering terms are used in the extracts below: *patch*, *user option*, and *version/product*. Patch and user options are adaptations specifically made for a single customer, whereas a version means the solution will be incorporated in the next release of the product and available for all customers.

Extract 7: In this extract local spread is illustrated in the context of the oil company (see Figure 2). The respondent talks about how the company in 2001 landed a big contract with the oil company, which was to purchase *Planner* for all PCs in its organisation (several thousand licenses). The context is the same as extract 2.

> [..] our main customer, Statoil, has spread ((the product)) throughout the organisation, and based on that we have received numerous requests for new functionality. We have agreed to do this as long as they pay for the development costs. And as a result more functionality has been added to the product.

The respondent explains how the multiple installations of the product in this organisation triggered numerous requests for new functionality. He explains that this was done through user an option/patch made for this particular customer. The respondent points to how this in turn led the company to make additional enhancements to the product for this company. The latter is an example of “global spread” that we elaborate below.

Extract 8: The conversation in this extract goes into more detail and is a follow-up to the discussion in extract 5. This developer’s view shows how a specific solution (available for one customer) can be made into a product available for all customers.

1. Interviewer1: Do you have other examples of customers initiating new functionality to the product?
2. Respondent2: Yes, we have done it for Kværner, ABB…[two large European engineering and consultancy companies]
3. Interviewer1: What sort of new functionality did they want?
4. Respondent2: I don’t remember. It was years ago. I know that when they bought the product they had specific demands that were originally not part of the product. But we wrote it in the contract as functionality they wanted.
5. Interviewer1: Ok, so it was a part of the contract?
6. Respondent2: Yes, they wanted it within a specific time period. Their requests were rather demanding regarding what they wanted us to do.

7. Interviewer1:	Was it an add-on made specifically for Aker Kværner?
8. Respondent2:	No, it became a part of the product. Yes, it started as a patch, what we call a user option.

Interviewer1 asks the respondents to give examples of customer-initiated product development. Respondent2 responds by naming two large European engineering and consultancy companies. Interviewer1 asks the respondent to specify what kind of new functionality was implemented. The respondent answers by making a reference to time *(“It was years ago”)*. This indicates that contributions from customers that are accepted for inclusion into the general product do not occur very often, and are mainly initiated by large customers. He explains how a major request for change led from “user option” to “patch” and eventually “*became part of the product*”. In the contract it is specified as a user option. The developers use the term “patch” to mean a plug-in to the product when it is run. Updates and fixes are incorporated this way. When a patch is integrated into the generic product code and made available for all customers it is “part of the product.”

The narrative illustrates global spread, an adaptation that becomes available to all customers. The transition from local spread to global spread, which in this case involved new licenses and local updates, is captured by the following abbreviated answer by Respondent2, in essence recapitulating the whole process in reverse chronological order *(“No, it became a part of the product. Yes, it started as a patch, what we call a user option”).*

In sum, the last two extracts taken together show there exists a path from adaptation to generalisation that is subject to multiple factors: local proposals for change, acceptance with or without payment, different stages for the artifact to pass through in the company (user option, patch, version) before it can be packaged into a generic product. In this process, both the customer and the company will benefit, despite the additional developmental work and expenditure involved, and the possible loss of owner rights to specific solutions.

GENERAL DISCUSSION AND CONCLUSIONS

In the two empirical sections we have described and analysed long-term development of the software, which we have called *generalisation*, and short-term development, which we have called *adaptation*. These two different but related processes are what constitute software product development in the company at a high level of abstraction. How they differ and how they relate give answers to the research questions that we brought up in the beginning of the chapter and summarise below.

From a CHAT perspective customer-initiated software development can be explained as *boundary crossing* and *co-configuration*. The boundary crossing is related to both the process of software development (with customers) and object transformation. Furthermore the former creates conditions for the latter. Therefore boundary crossing in this case is more than developer-customer collaboration; it is also about the migration of people and products among industries (from oil and gas to building construction). The contradictions are manifest in the processes related to adaptation and generalisation. They are parallel processes at different levels of development yet intertwined. This relationship must not be seen in a causal way, but as mutually dependent networked activities of people and software that lead to improved products and migration in multiple and diverse ways. Based on the success of the company over the past several years, we can say that the contradictions have been resolved and created cycles of expansion for the company.

The difference between the two levels of development is related to the timescales involved and in which direction the software as emerging objects goes. Adaptation at the most specific level is in most cases part of the short cycles of activities, but if they address increased specification and adaptability towards specific user-groups we can also talk about extended timescales on this level. Generalisation, however, is about long cycles of activities based on co-configuration between the customers and the company and mediated by the software (products and knowledge management support). Innovations are initiated as local adaptations to the products during a short development cycle and may be brought forward to all customers (at another time scale of development). We will now elaborate these findings along three main dimensions: from one instance to many, from Planner to MPX (Microsoft Project Extension), and from one application domain (oil and gas) to new application domain (building construction).

From one instance of the product to many: There is an intimate relation between short-term local development events and the long-term development of the product at large. Local development by the company and improvement requests and end-user tailoring by customers may end up in future versions of the product. It is those that are judged to be ‘good enough’ that will be pursued further; the others will be rejected or required to be paid for. We have identified stages (user option → patch → version) and multiple paths (good, possible, bad) along the route, which may lead to *aggregation* into a product. However, this process is not unique to our case. It has many things in common with, for example, open source software development and related initiatives such as outsourcing. In their efforts to distribute participation and organisation among multiple stakeholders distributed in time and space, multiple stages and paths to formalise ideas into stable intermediate forms become evident. However, this is more complex than combining existing methods in user-centered design (HCI) and software engineering. Further work is needed to develop a new method for customer-initiated software development along the lines suggested.

The difference between customer-initiated development and the related initiatives is manifest in that there are two types of activity systems involved and they have different objects and they contribute to development in different ways. One (customer activity) is to use the tools for domain-specific project planning, and the other is to further develop the project-planning tools and to make a profit doing so (developer activity). Furthermore the collaborators contributed in uneven (asymmetrical) ways to the development. The smaller stakeholder created an extension to the product developed by the larger stakeholder. This was the case for all of the organisations we reported from above.

From Planner to MPX (Microsoft Project Extension): We have to look further into the role of the products. They seem to have at least two very different functions. On one hand, during development they stand out as an object of activity (i.e., development of *Planner*). On the other, during domain-specific use, they are used as tools for project planning in customer organisations in the oil, gas, and building industries. By being aware of the double nature of the tool/object and the contradictions it entails, resolving the contradiction will be able to provide an explanation for the company's expansion. During the expansion the developer activity system and the customer activity system will have different objects. The customer objects will be oriented according to their business, which is different from the developers' business.

Furthermore, the products also play a third role, as part of the boundary crossing activities between the company and the customers during customer-initiated product development. Both the short and long-term cycles of development are connected to the products in their role as cultural tools to mediate specific development. They serve as a base on which to develop further extensions. This becomes visible in the data in how the respondents use the products to account for both short and long-term development. They combine factual descriptions and justifications in order to explain the expansion.

Finally, *emergence* also comes into play in this context. The idea behind the original tool *Project* and the subsequent development started 4-5 years prior to the founding of the company in 1997. The original product is in effect a template for all subsequent products, which new products are measured against. In other words *Project* is a predecessor artifact in both *Planner* and *MPX*, and it is visible to those who know these products. This is illustrated in Figure 3, which shows the company's understanding of the relationship between the three products. *Project* is a predecessor artifact in *Planner* and *MPX*. Each of them extends the functionality of *Project* in a user-oriented direction (ease of use with *Planner* and increased collaboration with *MPX*).

FIGURE 3 NEAR HERE

From one application domain to a new application domain: In the data the respondents describe the new market (building construction) in different ways. They describe it by what we have called *factual descriptions* from the data material. In addition they use *narratives* where they tell about cutting jobs in the oil and gas sector, migration of professionals to building construction, and the contract with Statoil to develop a new version of *Planner* (*MPX*) that triggered collaboration with Microsoft.

Together these changes mean that the object of activity for the company is transformed and that they go through expansive learning activities. It is important to note here that it is the long cycles of activities that can create expansive learning. On the other hand, it is the more or less serendipitous short cycles of activities that will be the source of the expansion. The relation between the short and long-term cycles of activities represents an empirical as well as theoretical problem in cultural historical activity theory. In this chapter we have found examples of it manifest as contradictions between *adaptation* and *generalisation* that could be resolved. However, our conclusions are tentative since it represents one case study. It should be seen as a hypothesis for further work and follow-up investigations.

Acknowledgement

The authors thank the members of the KIKK project at InterMedia, University of Oslo, Renate Andersen, Shazia Mushtaq, Damir Nedic, and Espen Olsen for contribution to ideas and software prototypes presented here. We also thank the company for giving us access to their employees and customers. Anne Moen gave us constructive comments throughout the project. A special thanks to Dave Middleton for detailed comments on two previous versions of this paper. The research is part of the KP-Lab project, financially supported by the European Commission, FP6-2004-IST-4 027490.

References

1. Andersen, R. and Mørch, A.I. (2009) 'Mutual Development: A Case Study in Customer-Initiated Software Development', in V. Pipek and M.B. Rosson (eds) *Proceedings 2nd Intl. Symposium on End-User Development (IS-EUD 2009),* Heidelberg & Berlin: Springer Lecture Notes in Computer Science (forthcoming).

2. Dodds, A.E. and Valsiner, J. (1997) The Personal and the Social: Mead's Theory of the 'Generalized Other'. *Theory & Psychology,* 7(4):483-503.

3. Ehn, P. (1988) *Work-Oriented Design of Computer Artifacts*, Stockholm: Arbetslivscentrum.

4. Engeström, Y. (1987) *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*, Helsinki: Orienta-Konsultit.

5. Engeström, Y. (1999) 'Innovative Learning in Work Teams: Analyzing Cycles of Knowledge Creation in Practice', in Y. Engeström, R. Miettinen and R-L Punamäki (eds) *Perspectives on Activity Theory*, Cambridge: Cambridge University Press.

6. Engeström, Y. Engeström, R. & Vähäaho, T. (1999) 'When the Center Does not Hold: The Importance of Knotworking', in S. Chaiklin, M. Hedegaard, M. and U.J. Jensen (eds) *Activity Theory and Social Practice: Cultural-Historical Approaches*, Århus: Aarhus University Press.

7. Fischer, G. (2007) 'Meta-Design: Expanding Boundaries and Redistributing Control in Design', in C. Baranauskas et al (eds) *Proceedings of Interact 2007 Conference (LNCS 4662)*, Heidelberg & Berlin: Springer Lecture Notes in Computer Science, 193–206.

8. Fischer, G., Lemke, A., McCall, R. and Morch, A.I. (1996) ‘Making Argumentation Serve Design’, in T.P. Moran, J.R. Carroll (eds) *Design Rationale: Concepts, Techniques, and Use*, Hillsdale, NJ: Lawrence Erlbaum.

9. Fischer, G., and Ostwald, J. (2001) ‘Knowledge Management: Problems, Promises, Realities and Challenges’, *IEEE Intelligent Systems*, 16(1): 60–72.

10. Gasser, L. (1986) ‘The Integration of Computing and Routine Work’, ACM *Transactions on Information Systems*, 4: 205–225

11. Ludvigsen, S.R., Havnes, A. and Lahn, L.C. (2003) ‘Workplace Learning across Activity Systems: A Case Study of Sales Engineers’, in T. Tuomi-Gröhn and Y. Engeström (eds) *Between School and Work: New Perspectives on Transfer and Boundary-Crossing*, Amsterdam: Elsevier Science, 292–310.

12. Mead, G.H. (1932) *The Philosophy of the Present*, ed. A.E. Murphy, Chicago & London: Open Court.

13. Mead, G.H. (1934), *Mind, Self, and Society*, ed. C.W. Morris, Chicago & London: The University of Chicago Press.

14. Mørch, A. (1996) ‘Adaptation through End-user Tailoring’, in M. Muhlhauser (ed) Special Issues in Object-Oriented Programming. *Workshop Reader of the*

10th European Conference on Object-Oriented Programming (ECOOP'96), Linz, Austria: Dpunkt Verlag, 43–52.

15. Mørch, A.I. (2003) 'Evolutionary Growth and Control in User Tailorable Systems', in N. Patel (ed.) *Adaptive Evolutionary Information Systems*, Hershey, PA: Idea Group, 30-58.

16. Nedic, D. and Olsen, E.A. (2007). 'Customizing an Open Source Web Portal Framework in a Business Context: Integrating Participatory Design with an Agile Approach', unpublished thesis, University of Oslo.

17. Nygård, K.A. and Mørch, A.I. (2007) 'The Role of Boundary Crossing for Knowledge Advancement in Product Development', in T. Hirashima et al. (eds) *Proceedings Int'l Conf. Computers in Education (ICCE 2007),* Amsterdam: IOS Press, 183–186.

18. Pollock, N. and Williams, R. (2008) *The Biography of the Enterprise-wide System or How SAP Conquered the World*, London: Routledge.

19. Roth, W.-M., and Lee, Y.J. (2007) '"Vygotsky's neglected legacy": Cultural-Historical Activity Theory', *Review of Educational Research*, 77: 186–232.

20. Scott, M.B., and Lyman, S.M. (1968) ‘Accounts’, *American Sociological Review*, 33(1): 46-62.

21. Simon, H.A. (1996) *The Science of the Artificial,* 3rd edn, Cambridge, MA: MIT Press.

22. Star, S. L., and Griesemer, J. R. (1989) ’Institutional Ecology, ”Translations” and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology’*, Social Studies of Science,* 19(3): 387–420.

23. Tuomi-Gröhn, T., Engeström, Y., and Young, M. (2003) *Between School and Work: New Perspectives on Transfer and Boundary Crossing,* Oxford: Elsevier Science.

24. Victor, B., and Boynton, A. C. (1998) *Invented Here: Maximizing Your Organization's Internal Growth and Profitability,* Boston, MA: Harvard Business School Press.

25. von Hippel, E. (2005) *Democratizing Innovation*, Cambridge, MA: MIT Press.

26. Åsand, H.-R., and Mørch, A.I. (2006) ‘Super Users and Local Developers: The Organization of End-User Development in an Accounting Company’, *Journal of Organizational and End User Computing,* 18(4): 1–21.

Figures

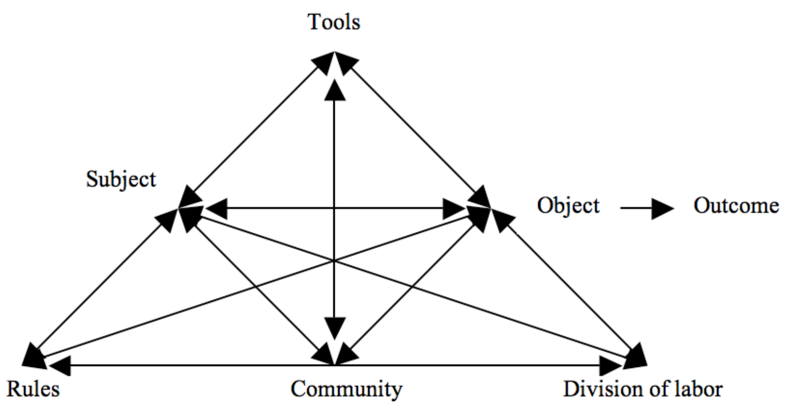

Figure 1: Graphical representation of activity system.

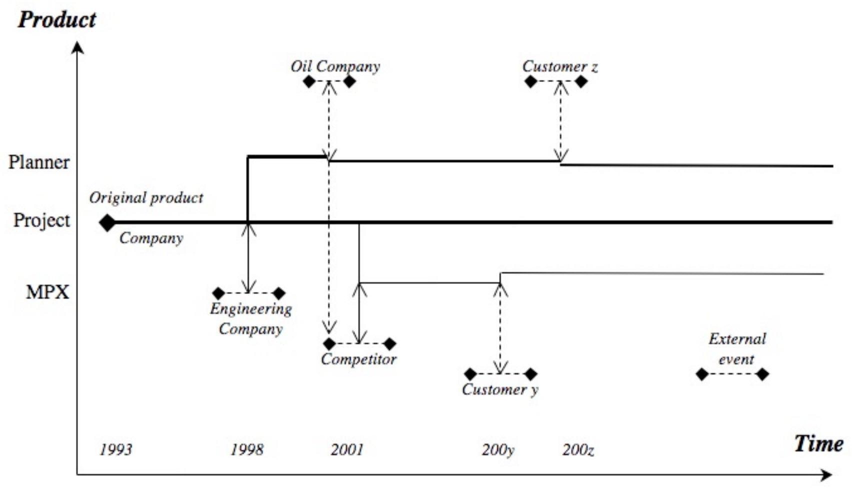

Figure 2: Y-axis shows products offered by company, depicting an expanding product line. X-axis shows external events that have lead to change. The vertical fixed double arrows mark “forks” in a product line and signify breaks (major changes). Incremental changes (stippled double arrows) lead to gradual improvement and continuation of products without forking.

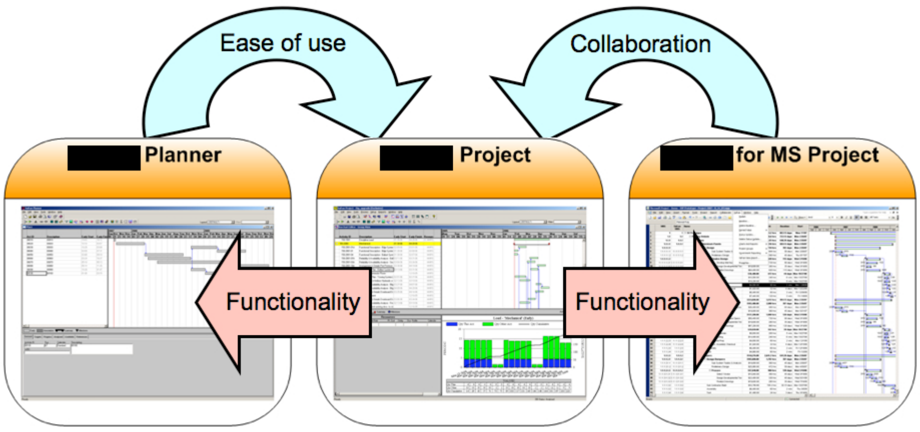

Figure 3: The company's illustration of the relationship between the three products: Planner, Project and MPX (company's extension to MS Project).