Gothenburg Studies in Informatics, Report 31, November 2004
ISSN 1400-741X (print), ISSN 1651-8225 (online)

Community-based customer involvement for improving packaged software development

Helena Holmström

Doctoral dissertation

Department of Informatics • Göteborg University
Viktoriagatan 13 • PO Box 620
405 30 Gothenburg • Sweden

Abstract

Noting the widespread use of virtual communities for interacting with customers, this thesis explores *the role of virtual communities for involving distributed customers in packaged software development* (PSD) and *the opportunities and challenges that are associated with this*. While the idea of involving customers in software development is not new, it is yet to gain momentum in PSD. Here, customers are distant and unknown — making traditional methods for customer involvement difficult to apply. Instead, packaged software developers use indirect links, such as intermediaries and customer surrogates, to communicate with customers. However, while these are cost-effective approaches for involving customers, there are problems associated with them. For example, filtering or distortion of information may occur. In this regard, virtual communities constitute an interesting approach for involving distributed customers more directly in PSD. In such communities, broad communities of interest, E.G., software customers, coalesce around products and services and instead of being involved only in idea generation, customers can co-create software, test software and provide each other with software support.

Conceptually, this thesis is based on a « knowing-in-practice » perspective, viewing community knowledge, i.e. situated knowledge as enacted in use by distributed software customers, as critical for improving packaged software. In accordance with this conception, knowledge creation processes are understood as expanding beyond the level of the firm, and as suggested in this thesis, PSD would benefit from utilizing also this knowledge. Methodologically, the interpretive case study is employed, using the hermeneutic circle as the guiding principle for the research process. Empirically, a Swedish computer game developer provides the context for assessing the role of virtual communities in PSD.

As a result of theoretical as well as empirical insights, this thesis presents community-based customer involvement as an approach for involving customers in PSD. In embracing opportunities as well as challenges, this approach views community knowledge as critical for improving PSD. For facilitating an understanding of the processes associated with community knowledge creation and transformation, the approach embraces a model for community use. In this model, community use is portrayed as a continuously ongoing interplay between the software firm and the software community. In this, knowledge creation and transformation processes are a result of commercial firm interests as well as voluntarily community participation. In understanding community use as portrayed in my model, there is the possibility to analyze how community knowledge is built, elicited and exploited from customer communities and hence, to what extent these can be used for involving customers in PSD.

Keywords: Packaged software development, software improvement, virtual communities, customer involvement, community knowledge, community-based customer involvement.

Language: English
Number of pages: 186

Gothenburg Studies in Informatics, Report 31, November 2004
ISSN 1400-741X (print), ISSN 1651-8225 (online)

Acknowledgements

In looking back at my time as a PhD student, I realize how fortunate I have been. Surrounded by inspiring and fascinating people, my thesis work has been an experience only a few people will have the privilege to enjoy. For this, I have many people to thank and while I always thought the acknowledgements would be the easy part to write, I now realize that there are no words for describing the gratitude I feel towards these people.

First, and always foremost, thank you Carl. You make me shine. Thank you for your patience, your support, your never-ending interest and your faith in me and in my ideas. Not many people are as lucky as I am — having a partner, a colleague, a best friend, a proof reader, a cover designer and a master of layout in one and the same person. In being by my side — no matter what — this thesis is as much yours as it is mine. I owe you so much.

Thanks are also due to my supervisor Ola Henfridsson and my co-supervisor Brian Fitzgerald for their support and devotion during these years. In encouraging as well as constructively questioning my work, you have contributed to the development of this thesis as well as to my personal development as a researcher. Thanks to you, these years will be years to remember and I will look back at my thesis with pride. Also, and in relation to the reading of my cover paper, I would like to thank Nancy Russo — a very special friend and colleague — and Geoff Walsham. The opportunity to have you read my manuscript provided me with feedback I wouldn't have gained elsewhere.

In being the place where I started my research, the Department of Informatics at Umeå University, and all the people working there, will always be very close to me. Thank you all for contributing to my research and for providing such a familiar and friendly atmosphere in which I always felt appreciated. Especially, thank you Charlotte Wiberg for getting me started on my thesis work — if it wasn't for you this thesis would never have been written in the first place. Thank you also Annakarin Nyberg for being a good friend and an inspiring colleague during the Daydream study. It was great fun working with you. For the latter part of my thesis work, the Viktoria Institute and the Department of Informatics at Gothenburg University have been important to me. Special thanks are due to all members of the Telematics Research Group for giving me excellent conditions for completing my thesis.

Besides colleagues — family and friends have made this process an enjoyable one. Thank you all for your love and support. Very special thanks to Anders Backman for 3D modelling one of the figures in the thesis and for great help in the cover design process and finalizing it for printing.

For funding — I owe my gratitude to the Center for Digital Business at Umeå University for the first two years, to Daydream Software in Umeå for support during the empirical part of my research, and to the Telematics Group at the Viktoria Institute for the final two and a half years of my PhD studies.

Finally, not many people include a horse in their acknowledgements. I do. With his ears flickering from curiosity, Pontus has listened to me going on about virtual communities and virtual community use and I can guarantee you — there is not a horse in this world as competent as Pontus when it comes to packaged software development.

Helena Holmström • Ödsmål • October 2004

Table of contents

1 Introduction

Traditionally, software development has been conducted within organizational contexts to satisfy organizational needs. Sometimes referred to as *custom* IS development (Sawyer, 2000), such development results in made-to-order software systems that are built for specific users that are identified before development begins (Keil and Carmel, 1995). Typically these systems, including software, hardware and people, are developed by either an organization's internal IT staff or by direct subcontract to a software house. Examples of such software would be legacy systems such as payroll systems, project planning and control systems, transaction processing systems, decision-support systems and office automation systems (Avison and Fitzgerald, 2003; Sommerville, 2001). In custom IS development, the idea of involving users in the development process is well recognized. This recognition is evident in studies on user involvement in different phases in the systems development process (Avison and Fitzgerald, 2003; Franz and Robey, 1986), in system implementation and use (Barki and Hartwick, 1994), and for system success (Ives and Olson, 1984, 1986; Tait and Vessey, 1988). To these ends, there are well-established development methods and techniques that support user involvement in the custom IS development process (see E.G., Avison and Fitzgerald, 2003; Checkland, 1981; Mumford, 1995).

However, we are experiencing a profound shift in how software is developed (Sawyer, 2000; 2001, Carmel and Becker 1995, Dubé, 1998). From being developed in-house and built by each user organization's own IT staff, I.E. custom IS, software is now to a greater extent developed by specialized software houses and sold as ready-to-install products, I.E. *packaged software*. Packaged software, also known as shrink-wrapped, commercial off-the-shelf or commercial software, refers to *all software sold as tradable products from a vendor, distributor or store, that are designed to be easily installed and to interoperate with existing system components* (Abts, 2002). As recognized by Carmel (1997), packaged software came about in the late 1960's as a result of an agreement between IBM and the United States Department of Justice to have IBM unbundle software from hardware, and has, since then, become an increasingly important form of information technology. In fact, the market for packaged software has grown to be the fifth largest industry in the US (Sawyer, 2000) and packaged software products are now widely used by both organizations and individual consumers.

As recognized by Sawyer (2000, 2001) and Carmel (1995), there are significant differences between custom IS development and packaged software development (PSD). Above all, while methods and techniques for user involvement[1] are common in custom IS development, these are yet to gain momentum in PSD. Here, customers are *distant* (Sawyer, 2000), I.E.

[1] As recognized by Keil and Carmel (1995) one significant difference between custom IS and packaged software development is that of terms used for denoting the software consumer. While « user » or « end user » is the common term in custom IS, « customer » has become the most common term used within the PSD domain. In accordance with this, I use the term « user » whenever referring to custom IS development and « customer » whenever referring to PSD. As a result of this, this thesis contributes with insights regarding *customer involvement*. For an additional discussion of the term, see also Grudin (1991).

geographically distant to the software developers, and *unknown* (Grudin, 1991), I.E. identified after development ends and the product is put on the market, making traditional methods for customer involvement difficult to apply. Instead, packaged software developers use a variety of indirect links, I.E. intermediated means, to communicate with customers (Sawyer, 2001). For example, packaged software developers build software to meet requirements gleaned from sources such as help-desk call-log analysis, market research, product reviews, surveys and user groups, of which continuous and direct customer contact is one of the least likely means (Keil and Carmel, 1995). While these techniques are indeed cost-effective for involving customers in PSD, there are several problems associated with the use of indirect customer links. For example, filtering or distortion of information may occur and from a PSD manager perspective, the use of indirect links is viewed as a significant factor in explaining why many PSD projects fail (Keil and Carmel, 1995).

Recognizing this problem, the question of *how* to involve customers more directly in PSD becomes relevant. Clearly, PSD can benefit from customer involvement and many are those who have highlighted customer knowledge as important for improving product development processes (Finch 1999; Von Hippel 1986). With profound experience and detailed knowledge of specific software products customers can be seen as possessing situated knowledge of the software and the particular situations in which it is used. Hence, customers — and the knowledge they possess — constitute a resource in the process of software development. What makes PSD particularly challenging is the fact that customers are distributed outside the traditional boundaries of the firm (Lee and Cole, 2003; Orlikowski, 2002). Thus, a critical issue for any packaged software developer aiming for customer involvement is to find techniques allowing for customers to be directly involved in development despite the fact that they are geographically distant to the developers.

In this, the emergence of new information and communication technologies has initiated a transformation of customer—producer relationships (Nambisan, 2002). With the advent of the Internet and Web-based technologies, distributed customer communities can now convene, interact and share resources extensively via electronic interfaces (Lee and Cole, 2003). As recognized by Nambisan (2002), the use of *virtual customer communities* for interacting with customers has become an interesting approach for facilitating innovation and knowledge creation processes in product development. In virtual communities, broad communities of interest, E.G., software customers, coalesce around specific products and services. Instead of being involved only in generating ideas for new products, customers can co-create products with firms, test products and provide each other with product support. In this regard, virtual communities can be seen as an interesting approach for involving distributed customers in knowledge creation processes necessary for product development. In such an approach, knowledge creation is seen as a process taking place not only within the boundaries of the firm but as a distributed process manifested in the interaction within virtual customer communities.

In recognizing virtual communities as a common — yet largely unexplored — approach for involving customers in product development, this thesis explores *the role of virtual communities* *for involving distributed customers in* PSD and the *opportunities and challenges that are associated with this*. Besides the call for empirical studies exploring this topic (Nambisan, 2002), my motivation for this is my belief that PSD processes could benefit from increased customer involvement and that indirect customer links could indeed be complemented with approaches allowing for more continuous and direct customer interaction.

In order to clarify this research question, a few words about its scope and possible limitations are appropriate. First, this study adopts a hermeneutic approach (Klein and Myers, 1999) which means that the findings are oriented towards the interpretation of human processes as they are understood and communicated within a specific empirical context (Patton, 2001). In this thesis, the research question concerns the role of virtual communities for involving customers in PSD processes. The research process is focused on mediating an understanding of PSD in order to make this phenomenon understandable for anyone interested in PSD and how it might be improved in terms of customer involvement.

Second, and as a consequence of the hermeneutic approach, the research question explored in this thesis is of an open character. The formulation of the problem is intended not to constrain the analysis but to make possible for different perspectives in the analysis. The empirical context that I explore — computer game development — involves a rich setting in which both opportunities and challenges are experienced. In my attempt to understand this empirical context, the alternative to delimit the research question would be constraining. While such delimitation would bring with it the possibility to streamline the study it would also risk making me less sensitive to the unique features that characterize this particular context.

Third, in using Walsham's (1995) classifications of different types of generalizations that can be made based on interpretive case studies, the findings of this thesis can be classified as *specific implications*. What I present is a detailed account of a specific case of community use. In terms of generalizability, generative mechanisms identified for phenomena in the social sciences should be viewed as « tendencies » which are valuable in explanations of past data but are not wholly predictive for future situations (Walsham, 1995). Therefore, the generalizations discussed in this thesis should be seen as explanations of a particular phenomenon derived from empirical interpretive research, which may be valuable in similar organizational contexts. Below, the thesis structure is outlined in terms of the papers that were selected to be included in the thesis.

Thesis structure

This thesis includes a cover paper and a collection of six individual papers. In the cover paper, my intention is not only to synthesize the research presented in each paper, but also to complement the discussion in the papers with knowledge that has emerged during the research process but, for different reasons, have not been included in the papers.

The cover paper is divided into six sections. Following the introduction, section two provides the background for my research. In identifying related research, this section provides an understanding of the area on which I build upon, as well as the area to which I wish my research to contribute to. In section three, my methodological choice of the interpretive case study is presented as well as the empirical setting in which this research was undertaken. Section four provides the theoretical perspective employed in the thesis. Here, an understanding of software development as a knowledge intensive activity is presented and virtual communities are outlined as enablers for the creation and transformation of knowledge inherent in distributed customer communities. On the basis of continuous transitions between theoretical concepts of « knowledge », « practice » and « community », and empirical insights gained throughout my research process, a conceptual model for understanding community use is presented, I.E. the community use model. In identifying environmental conditions as well as internal knowledge creation and transformation processes, the model provides an understanding of community knowledge as pivotal for improving PSD. Section five presents my research contributions in terms of a community-based approach to customer involvement. Finally, section six concludes the cover paper.

Following the cover paper is the collection of six papers that constitute the thesis. The papers are included in the same order as they were written. Due to the publication process of the first paper, however, it appears as if it was written after paper number two, although this was not the case. In the collection, there are two published journal papers, one submitted journal paper and three conference papers. Three of the papers are co-authored with my supervisor Ola Henfridsson, one is co-authored with my co-supervisor Brian Fitzgerald, and on the remaining two I am the single author. The six thesis papers are listed below.

Paper 1	Henfridsson, O., and Holmström, H. (2002). Developing E-commerce in Internetworked Organizations — customer involvement throughout the value chain in the case of the online computer game Clusterball. DATA BASE — Special Issue on Developing E-Commerce Systems, Current Practices and State-of-the-Art. VOL. 33, NR. 4, PP. 38-50.
Paper 2	Holmström, H. (2001). Virtual Communities as Platforms for Product Development — an interpretive case study of Customer Involvement in Online Game Development. *In Proceedings of* ICIS *2001*, *(22nd International Conference on Information Systems),* December 16-19, New Orleans, LA, USA.
Paper 3	Holmström, H., and Henfridsson, O. (2002). Customer Role Ambiguity in Community Management. *In Proceedings of* HICSS *35 (35th Hawaii International Conference on System Sciences),* January 7-10, Big Island, Hawaii.
Paper 4	Holmström, H. (2003). The Distributed Nature of Software Development — a comparison of three development approaches. *In Proceedings of* PACIS *2003 (Pacific Asia Conference on Information Systems),* July 11-13, Adelaide, Australia.

Paper 5 Holmström, H., and Fitzgerald, B. (forthcoming). Virtual Community Use for Packaged Software Maintenance. *Accepted for publication in the Journal of Organizational Computing and Electronic Commerce* — Special Issue on « Virtual Communities and Personalization in E-commerce ».

Paper 6 Holmström, H., and Henfridsson, O. (submitted). Improving Packaged Software Through Online Community Knowledge. *Submitted to an international* IS *journal.*

2 Software development

Software development, and the way in which it is conducted has for a long time been a core interest within the field of information systems (IS). Software development methods (Fitzgerald, 1996, 1997; Avison and Taylor, 1997; Nandhakumar and Avison, 1999), software development tools and techniques (Avison and Fitzgerald, 2003) and software development environments (Holmström, 2003) keep fascinating both researchers and practitioners, and due to continuous alteration, there are reasons to believe that this field will keep fascinating us also in the future.

A common way of characterizing software development is that of a series of sequentially organized phases such as *strategy*, *feasibility*, *design*, *programming*, *implementation*, *use* and *maintenance* (Clegg ET AL, 1997). Here, as well as in the software engineering perspective as presented by Sommerville (2001), the idea is that software development comprises a series of well-defined phases in which specific activities are performed. Each phase operates with a defined notation and will often result in a prescribed artifact, such as a design document or a program (Baskerville and Pries-Heje, 2001). In systems development this sequential model is frequently referred to as the systems development life cycle (SDLC) or the waterfall model (Avison and Fitzgerald, 2003; Sommerville, 2001), representing a linear approach to software systems development. This model consists of six phases that together constitute the steps necessary to develop a system that is tested, implemented and evaluated in relation to a system specification (Avison and Fitzgerald, 2003). In being a model that has been around since the late 1960s, the SDLC is known as a well-tried and tested approach that has, in one way or another, influenced most software development methods that are being used today (Avison and Fitzgerald, 2003).

In terms of *user involvement*, there are well-established methods for this within the field of software development (see E.G., Checkland, 1981; Mumford, 1995; Avison and Wood-Harper, 1986; Avison and Fitzgerald, 2003). In the SDLC, several attempts are made to involve users in the development process. First, there is a user specification in which users specify their requirements of the system. This is handed over to the system developers in the beginning of the development process and is then used throughout the process in order to develop a system according to initial user requests and requirements. Second, there is the opportunity for users to review progress at the end of each phase in the development process. However, while the SDLC — and the idea of linear systems development — is still significant in environments in which requirements are well understood, its application in more complex environments has shown to be problematic. The principal weakness is the underlying assumption that a solution can be achieved at through a sequence of phases. This implies that later phases depend on the successful completion of earlier phases, something that according to Henson and Hughes (1991) requires « perfect foresight ». Furthermore, researchers have noted that systems development very seldom is an orderly systematic process in which developers complete one task before moving on to another (Fitzgerald, 1997). Above all, despite the attempt to involve users in the development process, the linear model of software development suffers from its rigid structure and inflexible nature and hence, is limited in its ability to adapt to changing user needs and requirements. As recognized by Clegg ET AL (1997), knowledge that users possess of the detailed workings of the application domain, how it works and how it could work, are given less significance than more technical concerns. Often, users' knowledge is incorporated only by capturing their requirements during the early feasibility study, or through some form of acceptance testing during the later stages of implementation. Bearing in mind the recognition of users' having difficulties with articulating their requirements before the development process has begun (Nuseibeh and Easterbrook, 2000), and the fact that changes during the later stages of the development process is often considered inconvenient (Avison and Fitzgerald, 2003), there is no surprise that linear systems development, as that reflected in the SDLC, is often associated with user dissatisfaction and low user acceptance rates (Avison and Fitzgerald, 2003).

As a result of the limitations identified in relation to user involvement in the SDLC, numerous methods and techniques have been developed to better cater for this. For example, there are the socio-technical approaches recognizing "*...the interaction of technology and people and produces work systems which are both technically efficient and have social characteristics which lead to high job satisfaction*" (Mumford, 1983). In addition, there is the entire field of Participatory Design (PD) and contextual design (Beyer and Holtzblatt, 1998) representing human-oriented approaches towards systems development in which the people destined to use the system play a critical role during the development process. Recognized not as a single theory or methodology, but as a rich and diverse set of perspectives and experiences, PD aims at establishing a more cooperative process of technology development in which the gulf between developers and users can be bridged (Grønbæk ET AL, 1993). The approach can be seen as a response to the recognition that traditional software development methods often fail in involving users and consequently don't comply very well with users' needs and requirements (Grudin, 1993). As claimed by Jones (1988) "*...we must recognize that the « right » requirements are in principle unknowable by users, customers, or designers at the start*". Thus, techniques must be developed to make possible for user involvement during the software development process. In PD, it is believed that knowledge about users' practices and environments can only be obtained through close cooperation, and that a stronger focus on the development *process* will also result in better software *products* (Grønbæk ET AL, 1993). Common techniques as advocated in PD are for example language and organizational games (Ehn, 1993), mock-ups (Ehn, 1993; Holtzblatt and Jones, 1993), scenario-building (Holtzblatt and Jones, 1993) and user workshops (Greenbaum and Halskov Madsen, 1993).

However, despite methods and techniques for user involvement these are often difficult to apply in relation to certain software development environments or certain software products. As recognized in several studies within the field of IS (Sawyer, 2000, 2001; Carmel and Becker 1995; Dubé, 1998), there is an ongoing shift in how software is made. Rapidly changing technology, ever-shorter product life cycles, and ever-increasing competition exert pressure for prices on software products to go down and quality to go up (Dubé, 1998). Under these conditions, reducing time from idea to market becomes a fundamental competitive strategy (Carmel, 1995) and accordingly, new development methods emphasizing development productivity rather than process rigor arise. Today, software firms use methods such as eXtreme Programming (XP), Dynamic Systems Development (DSD) and Feature-driven Development (FDD) to deliver business value quickly, while also accommodate changing user requirements (Abrahamsson ET AL. 2003). Also, the forms of software products are changing. According to Sawyer (2000), there are at least three different forms of software. First, *custom* IS represents made-to-order software systems that are built for a particular user organization. Here, development is typically an internal business and the user organization is closely involved throughout the development process. Second, and as a growing segment, *packaged software* represents ready-to-install software products intended for mass use of customers distributed all over the world. Here, development is done by specialized software firms and due to the global distribution of customers, these are less involved in the development process. Third, and as a hybrid form, *applications with large packages*, such as for example enterprise resource planning (ERP) systems, represents software systems that are sold to large organizations but require extensive post-purchase tailoring to meet the specific needs of each customer organization. The move to purchasing a software application an then tailoring it to meet specific organizational needs reflects a « hybrid » response to the « build-versus-buy » decision as reflected in custom IS and packaged software. Further, both custom IS and packaged software, and its hybrid, differ from *embedded software*. Here, software is typically written to the hardware, making hardware and software intertwined and bundled. Of course, and as recognized by Sawyer (2000), the boundaries distinguishing these forms of software are blurred. Not the least, the hybrid form of ERP software purchase and subsequent tailoring show that there are indeed differences between different types of software that could still be classified as packaged software. Also, custom IS has undergone significant change in both development methods (Little, 2004) and development tools (Clegg ET AL, 1996).

In this thesis, I focus on *packaged software,* not in terms of large business solutions serving multiple purposes (such as application packages), but in terms of software solutions serving limited purposes for individual software consumers (such as a computer game or a development platform). While this research might prove relevant also for hybrid forms of packaged software, this is not an intended goal in this thesis. Below, *custom* IS *development* is briefly outlined as what has historically been the main concern within the field of IS development, and what most software developers have been accustomed to. Following this, *packaged software development* is presented as one specific, and growing, branch of software development that is significantly different to custom IS in terms of both development environment and development practice. In contrasting custom IS and PSD, there is the recognition of customers as less involved in PSD. Challenged by this, the remaining part of my thesis focuses on the role of virtual communities for involving distributed customers in PSD, and the opportunities and challenges that are associated with this.

2.1 CUSTOM IS DEVELOPMENT

Traditionally, software has been developed as a part of larger information systems, most often intended for organizational use. According to Avison and Fitzgerald (2003), these systems include many interacting components, such as people (E.G. analysts and business users), objects (E.G. computer hardware devices) and procedures (E.G. activities suggested by an IS development methodology) that are all important to the overall system development process. The characteristic feature of these systems is that they are customized for one particular organization and hence, intended to support the specific behavior and structure of that organization.

Typically, custom IS systems, or bespoke products (Sommerville, 2001), are systems that are developed by either an organization's internal IT staff, or by subcontract to a software house (Sawyer, 2000). In other words, custom IS are *made-to-order systems that are built for specific users in a specific organizational setting.* This definition of custom IS includes for example payroll systems, project planning and control systems, conferencing systems, transaction processing systems, decision-support systems and office automation systems (Avison and Fitzgerald, 2003; Sommerville, 2001). Also, this definition of custom IS includes most government work. As recognized by Sawyer (2000), software development of the US Department of Defence is typically custom IS development.

In terms of development environment, custom IS is characterized of much of what has been described in the previous section on traditional software development (section 2). Here, the development process is mature with well-established development methods describing separate phases such as for example design and development, reflecting an engineering view on the development process. Also, and due to limitations in the SDLC — which has to a great extent influenced the development methods applied in custom IS today — there are numerous techniques for user involvement. From being a shortage, user involvement is today a central belief within custom IS development (Sawyer, 2000). For example, there are techniques such as prototyping (Avison and Fitzgerald, 2003; Baskerville and Wood-Harper, 1998) and contextual inquiry (Beyer and Holtzblatt, 1998), and methods such as Effective Technical and Human Implementation of Computer-based systems (Mumford, 1995) and Soft Systems Methodology (Checkland, 1981) emphasizing user involvement during the software development process.

However, while custom IS will always be an important strand of software development practice, and certainly what many future software developers will be acquainted with, there is an on-going shift in how software is made and what type of software products that are produced. To a larger extent we are experiencing ready-to-install products intended for a mass market of distributed customers, I.E. packaged software. Below, this type of software, as well as the conditions under which it is developed, is outlined.

Packaged software, also known as shrink-wrapped, commercial off-the-shelf or commercial software, refers to *all software sold as tradable products from a vendor, distributor or store, that are designed to be easily installed and to interoperate with existing system components* (Abts, 2002). Furthermore, Carmel (1997) describes this type of software as *competitive*, both domestically and across national frontiers. As recognized by Carmel (1997), packaged software came about in the late 1960's as a result of an agreement between IBM and the United States Department of Justice to have IBM unbundle software from hardware, and has, since then, become an increasingly important form of information technology. In fact, the market for packaged software has grown to be the fifth largest industry in the US (Sawyer, 2000) and packaged software products are now widely used by both organizations and individual consumers.

As recognized by Sawyer (2000), many of the largest packaged software firms are well-known (E.G., Adobe, Microsoft, and Oracle) and examples of packaged software products are operating systems such as Microsoft Windows and Mac OS, desktop publishing software such as Adobe CS and Quark Express, database programs such as Microsoft SQL and Oracle, and computer games such as Blizzard's Warcraft and Sony's Everquest. However, the largest growth in packaged software is applications, as different from system software, with large packages of which enterprise resource planning (ERP) software (E.G., SAP), is the fastest growing segment. This type of software often requires extensive tailoring to meet the specific needs of, for example, an organization. This kind of modification is also true for other large business solutions such as document management software (E.G., DOCS Open and Groupwise).

In terms of development environment, the PSD process is considered immature in that formal development methods are not used to the same extent as in custom IS and, instead of separate phases, the development process is often characterized more of integrated design and development (Sawyer, 2000). In considering customer involvement, there are certain difficulties associated with PSD. First, customers are *distant* (Sawyer, 2000), and as a result of this, less involved in the PSD process. Looking at successful packaged software products such as Microsoft Windows, Adobe Pagemaker or games such as Warcraft and Everquest, customers can be found all over the world. As thrilling as this might seem, this fact makes it difficult to use conventional techniques for user involvement as advocated by, for example, traditional systems development (Avison and Fitzgerald, 2003) or participatory design (Greenbaum and Kyng, 1991; Namioka and Shuler, 1993). While these techniques succeed in capturing both individual expectations and organizational needs, activities such as user workshops, role playing and mock-ups are typically location-dependent activities and hence, do not resonate well in the distributed environment of PSD. Instead, PSD uses indirect links such as *support lines,* I.E. the unit that helps customers with day-to-day problems, *surveys*, I.E. textual surveys administered to a sample of customers, *user groups*, I.E. customer groups convening periodically to discuss software usage and improvements, *trade shows*, I.E. customers are exposed to prototypes and asked for feedback at a trade show, and *marketing* *and sales*, I.E. firm representatives meet customers to listen to suggestions and needs (Keil and Carmel, 1995), for involving customers in the development process. However, while these approaches make possible for a cost-effective customer-developer interaction in distributed settings, this interaction is often mediated through intermediaries or customer surrogates (Keil and Carmel, 1995) and as a consequence, difficult to rely too heavily upon. In this discussion, intermediaries are seen as entities situated between customers and developers, while customer surrogates are entities that are not true customers but are treated as such for the purpose of gathering requirements and feedback. Realizing this use of indirect customer-developer links (Keil and Carmel, 1995), it is fair to say that packaged software customers have often played a largely passive role with firms employing a range of structured inquiry mechanisms to import customer knowledge. Also, logistical and economic considerations force firms to involve only a minority of customers in these inquiries (Nambisan, 2002), and while recent research suggests groupware systems for interacting with distant customers (Tuikka and Salmela, 1998; Fukushima and Martin, 1998) these systems are difficult to apply in PSD where the products are intended for a mass market of which representative customers are difficult — and sometimes impossible — to identify in advance (Keil and Carmel, 1995).

Second, customers are *unknown* (Grudin, 1991), and often remain unknown until the software is marketed. Certainly, all software development begins with some idea of its intended customers, for example, if they are already existing customers or a new market. But still, the development situation of packaged software is clearly separated from the use context and the uncertainty of the eventual customer is an important facet of product development, as well as the unexpected fates of many products (Grudin, 1991). In contrast to custom IS development, PSD is targeted to a mass market of distributed customers. As well as organizations are characterized according to the work routines that are carried out, organizational members can be characterized according to the context of which they are part. Thus, custom IS development benefits from the fact that users are part of the same organizational context and have a common culture to which they can relate. In PSD, on the other hand, customers are not part of a coherent use context, and there exists little data material for characterizing customers and for developing representative use models. Here, there are no formal work patterns, no organizational processes and no organizational goals available to facilitate the process of understanding and representing customers. Consequently, customers cannot to the same extent be represented in terms of profession and professional domain, and the overall context in which the software is to operate — and the purpose for which it is used — might not be shared but instead different for every single customer. As recognized by McDonough ET AL. (2001), the process of understanding customer needs and requirements is no longer a matter of identifying needs of a relatively homogenous group, as for example of users within an organization. Instead, it requires an understanding of globally distributed customers who speak different languages, who have different cultural beliefs and who may not be able to ever meet physically with each other or with the software developers. While, for example, scenario-building and rich pictures (Checkland, 1981) aim at capturing the complex relationships within an organization, PSD cannot to the same extent benefit from a shared use context, and the developers can in most cases only guess which customer community and what types of customers that will use the products (Divitini ET AL. 2000).

2.3 SOFTWARE FIRM ENVIRONMENTS

As recognized in the discussion above, there are a number of characteristics that distinguish the PSD environment from the custom IS development environment. Based on two case studies, Sawyer (2000) identifies four distinguishing characteristics in terms of (1) industry, (2) software development, (3) cultural milieu, and (4) teams (TABLE 1).

Characteristics	*Packaged Software Development*	*Custom IS Development*
Industry	Time to market pressures Success measure: profit, market share, mind share	Cost pressures Success measures: satisfaction, user acceptance, ROI
Software Development	Line positions User is distant and less involved Process is immature Somewhat integrated design and development Design control via coordination	Staff positions User is close and more involved Process is mature Separated design and development Design control via consensus-building
Cultural Milieu	Entrepreneurial Individualistic	Bureaucratic Less individualistic
Teams	Less likely to have matrix/project structure. More likely to be self-managed Involved in entire development cycle More cohesive, motivated, jelled Opportunities for large financial rewards Small and collocated Share a vision of their product(s)	Matrix managed and project focused People assigned to multiple projects Work together as needed Salary-based Grow larger over time and tend to disperse Rely on formal specifications and formal documents

TABLE 1. Summary table of differences between packaged software and custom IS software firm environments (Sawyer, 2000)

First, *packaged software industry* is dominated by time pressures. To break new ground in bringing new and innovative products to the market is critical to create return on the investments done by either venture capital or state-supported incubator money. Also, the success of the packaged software industry's products is measured by profit, and to achieve this, there is the challenge of either developing a large installed base or to create new market opportunities.

Second, *packaged software development* is conducted by developers that often hold line positions which make their needs central to the performance of the organization (Sawyer, 2000).

Instead of being part of corporate staff and serve supporting roles, as is often the case for developers within custom IS development, packaged software developers are production mechanisms and hence, those who generate revenue. Also, in the PSD process there is a product focus which cannot to the same extent be seen within custom IS development. Furthermore, the process is immature and there is a distant relationship to customers since customer needs and requirements are most often filtered through intermediaries. This means that formalized software development methods as well as methods for customer involvement, are not used to the same extent in PSD as is the case in custom IS development. Rather, the PSD process is highly iterative, flexible and constantly evolving and while there are attempts to involve customers, these are often restricted to beta testing or demonstrations.

Third, *cultural milieu*, in terms of ideas, values and shared norms, of PSD is entrepreneurially-oriented and individualistic (Sawyer, 2000), and the hierarchical and bureaucratic structures that can often be found within custom IS are not a characterizing feature here.

Finally, *packaged software development teams* are typically small and co-located. As recognized by Sawyer (2000), PSD teams tend to be stable and remain committed to a product over several versions or releases. Consequently, they often work together for long periods of time and define goals over prolonged periods. Also, motivation is often manifested in financial rewards such as stock options and bonuses that can provide a lucrative bonus for developers in PSD firms.

As can be seen in TABLE 1, there are fundamental differences between PSD and custom IS environment. However, while differences in terms of industry, cultural milieu and development teams are indeed both interesting and challenging, and certainly aspects that influence the PSD process, my focus in this thesis is the fact that customers are recognized as distant and less involved in the PSD process (Sawyer, 2000). Recognizing this, and keeping in mind the apprehension of customers as having many of the best ideas for product improvement (Finch, 1999; Von Hippel, 1986), there is the need for approaches to involve customers also in PSD. In involving customers, there would be the opportunity to exploit knowledge inherent in the customer community and improve PSD products and processes in better accordance with this. In this thesis, *virtual communities* are explored as one such approach. In being conceptualized as the association of community members and the enabling electronic medium, virtual communities are constituted by interesting structures that provides benefits by supporting interpersonal relationships, encouraging knowledge sharing, allowing for quick access to information and enabling collective action such as, for example, software development (Butler, 2001).

3 Research design

This section presents the research design that was employed in this thesis. In the first part (section 3.1), the *interpretive research approach* and the particular choice of the *interpretive case study* is outlined to create an understanding for the underlying philosophy that was adopted in this research and the assumptions that follow this. In addition, a set of principles for conducting and evaluating interpretive research (Klein and Myers, 1999) is presented as background for a later discussion of how the research reported in this thesis was conducted and hence, how it can be evaluated. In the second part (section 3.2), the *empirical context* is described to provide the reader with an insight in this particular case setting and why it was chosen for this study. In the third part (section 3.3), my *research process* is outlined as well as the way in the principles, as presented by Klein and Myers (1999), were adopted. It is my intention that this chapter will provide the reader with a good understanding of how this research was conducted and the way in which my empirical field work was carried out to explore the particular phenomenon of study.

3.1 THE INTERPRETIVE RESEARCH APPROACH

From being almost non-existent within the IS research community at the beginning of the 1990s, the interpretive research approaches are now accepted as part of the mainstream of the information systems research community (Markus, 1997). For the interpretive researcher, the foundational assumption is that most of our knowledge is gained through social construction such as language, consciousness, shared meanings, documents, tools and other artifacts. Furthermore, this type of research does not predefine dependent and independent variables, but focuses on the complexity of human sense making as the situation emerges (Klein and Myers, 2001). Here, the understanding of human thought and action in social and organizational contexts is of primarily interest (Klein and Myers, 1999). In the words of Walsham (1993, P. 4-5), interpretive methods of IS research are "*...aimed at producing an understanding of the context of the information system, and the process whereby the information system influences and is influenced by the context*" and the interpretive researcher face the challenge of understanding a phenomenon through the different meanings that people assign to them.

While there are different types of interpretive research approaches (Mingers, 1984), they all differ from the positivist research tradition in terms of epistemology, I.E. the nature of knowledge claims, and ontology, I.E. the nature of reality. Following the positivist tradition, facts and values are distinct and it is considered that scientific knowledge consists of facts (Walsham, 1995). Generally speaking, IS research can be classified as positivist if there is evidence of formal propositions, quantifiable measures of variables, hypothesis testing, and the drawing of inferences about a phenomenon from a representative sample to a stated population (Orlikowski and Baroudi, 1991). In contrary to this, interpretive research starts out with the assumption that (1) facts and values are intertwined and hard to disentangle and that both are involved in scientific knowledge I.E. « non-positivism », or that (2) scientific knowledge is ideological and inevitably conducive to particular sets of social ends I.E. « normativism » (Archer, 1988; Walsham, 1995).

In terms of ontology, I.E. the nature of reality, the positivist tradition considers reality as existing independently of our construction of it, I.E. external realism. The interpretive researcher, on the other hand, views reality as an intersubjective construction of the shared human cognitive apparatus, I.E. « internal realism », or as « subjective idealism » where each person is considered to construct his or her own reality (Walsham, 1995). Here, the assumption is that access to reality, given or socially constructed, is only through social constructions such as language, consciousness and shared meanings (Klein and Myers, 1999). So, what does it mean then to adopt an interpretive research approach when studying an IS phenomenon? As recognized by Henfridsson (1999), this brings with it the privilege of developing research within an established tradition of research. Thus, instead of having to lay the foundations for conducting such research, the interpretive researcher can concentrate on developing and refining the contents of his work within an already established tradition. Also, it means that there exist explicit criteria for conducting and evaluating such research. Klein and Myers (1999) have made such criteria explicit for interpretive field studies and present the hermeneutic circle as the fundamental principle of interpretive research. Below, the particular choice of the interpretive case study as a research method is outlined as well as the hermeneutic circle as the fundamental principle for conducting and evaluating this type of research.

3.1.1 The interpretive case study

As recognized by Yin (2003) the case study is but one of several ways of doing research. Other ways include experiments, surveys, histories and the analysis of archival information. While Yin adopts an implicitly positivist stance in describing case study research, there are many points of agreements between the positivist and interpretivist approaches to case studies. As pointed out by Walsham (1995) — in representing the interpretive school, any interpretivist would accept Yin's (2003) view that case studies are the preferred research strategy *"…when a « how » or « why » question is being asked about a contemporary set of events over which the investigator has little or no control"* (Yin, 2003, P. 9). Furthermore, (Yin 2003, P. 13) defines the scope of the case study as *"an empirical inquiry that (1) investigates a contemporary phenomenon within its real-life context, especially when, (2) the boundaries between phenomenon and context are not clearly evident".*

Clearly, the case study research method is particularly well-suited to IS research, since the object of this discipline is the study of human actions and interpretations surrounding the development and use of computer-based information systems (Walsham, 1995), and where multiple sources of evidence has to be collected and abstracted to get plausible answers to the research questions (Gillham, 2000). Indeed, the IS literature contains reports from a significant number of interpretive case studies, covering a wide range of different topics and issues (see E.G., Markus, 1983; Suchman, 1987; Orlikowski, 1992: 2002, Walsham, 1993).

As a result of the emergence of interpretive research as an important and accepted approach to research within IS, as well as due to my own interest in the way of conducting research as advocated by this tradition, the case study approach was deployed in the study reported on here. In exploring virtual communities and the potential of these in terms of customer involvement, the interpretive approach in general, and the interpretive case study in particular, provided me with a set of methods and techniques for elaborating on the topic chosen. Below, a set of principles for conducting and evaluating such studies are outlined

(Klein and Myers, 1999). Following this, the empirical context in which my empirical work was conducted is described.

3.1.2 Principles for conducting and evaluating interpretive case studies

As the fundamental principle for conducting and evaluating interpretive research, Klein and Myers (1999) present the *hermeneutic circle*. This meta-principle encompasses the whole process of interpretation and asserts that understanding stems from seeing the interrelation between wholes and parts of a phenomenon (Klein and Myers, 1999). The idea is that we come to an understanding of a complex whole from preconceptions about the meanings of its parts and their interrelationships. In this process, the interpretation moves from a precursory understanding of the parts to the whole and from a global understanding of the whole context back to an improved understanding of each part.

In this thesis, my aim was to understand *the role of virtual communities for involving distributed customers in* PSD, and the *opportunities and challenges that are associated with this*. For such an understanding, the hermeneutic circle implies that one needs to iterate between parts and wholes, while reflecting upon how the pre-understanding affects the researcher's understanding. With regard to my phenomenon of study, this means that a hermeneutic process might consist of me developing an understanding by using my pre-understanding to go back and forth between *parts* such as empirical details as experienced in the empirical case and *wholes* such as literature on internetworked organizations and customer involvement in software development.

Besides the fundamental principle of the hermeneutic circle, Klein and Myers (1999) introduce six additional principles which are all useful for conducting and evaluating interpretive research (TABLE 5 in section 3.3.2). In being consistent with a considerable part of the philosophical base of literature on interpretivism, these principles are seen as guidelines from which interpretive research can be better judged and assessed. In applying these principles it is argued that a theoretical base is established which would otherwise have to be derived by each interpretive researcher. Also, these principles allow for researchers to defend their work to principles that are firmly grounded in at least one major direction of interpretive philosophy, I.E. the hermeneutic tradition. Finally, the introduction of a set of principles encourages researchers to consider each one of the principles systematically instead of otherwise neglecting important aspects of their work (Klein and Myers, 1999). In working as guidelines for the conduct and the evaluation of interpretive research, the principles as presented by Klein and Myers (1999) were considered in the research reported on in this thesis. Below, the empirical context in which this research took place is outlined (section 3.2). Following this, my research process and the way in which Klein and Myers (1999) principles were considered in this study is presented (section 3.3).

3.2.1 Daydream Software

The case study reported on in this thesis was conducted at Daydream Software (referred to as « Daydream »). Daydream is a Swedish computer game developer with its headquarters in Umeå. By the time for this study (2000—2002) the company employed 65 people ranging from administrative personnel, marketing people and executives to software developers, web designers and graphical designers. Overall, the company consisted of young people and Daydream's intention was to create innovative computer games that attracted people in all ages and from all over the world. Mainly, the focus was on sports games and strategic games and the general attitude was to avoid violence in the games.

As a starting-point for this study there was the development of Clusterball — Daydream's third production. Earlier, the company had released two computer games, Safecracker and Traitors Gate, which had both been very successful in Europe as well as in the US. While these games were sold as packaged software that could be purchased from any vendor world-wide, Clusterball was different in that here Daydream's intention was to develop a game that, in addition to being sold by vendors, also could be downloaded and experienced from the Internet. The main motivation for this was the opportunity to attract even more customers as well as the challenge of entering a new medium for launching the game. Clusterball — a game in which you fly around in ships trying to collect and steal balls from other players — would be their first multiplayer game to be downloaded and experienced from the Internet. In relation to the game there was an electronic payment system as well as a customer relationship management database (CRM database) in which all players could register and through which Daydream could keep track of its customers. The expectations on Clusterball were high, and when entering the study in early 2000, I could sense a feeling of excitement throughout the company in developing what would become a groundbreaking product in many respects (for a more detailed description of Clusterball see section 3.2.3).

In developing Clusterball, Daydream recognized customer knowledge and skills as critical to the development process. In looking back at earlier products that were static in the sense that once they were released customers could no longer influence them, Daydream wanted to make possible for customer involvement during — but also after — development. In this way Daydream wanted to enlarge the possibility to adjust the product according to customer needs and requirements. To achieve this, a *virtual community* was created on the Clusterball website. Here, customers from all over the world could meet and by using electronic forums these customers got the possibility to discuss every detail of the game as well as to sign up for tournaments and other activities organized by Daydream and other community members (for a more detailed description of the Clusterball community see section 3.2.4). In using virtual community functionality, there were opportunities for knowledge-sharing among Clusteball customers and by taking part in their discussions, Daydream developers could learn from customers and the knowledge they possessed.

3.2.2 *Motivation for choice of research site*

For me as a researcher interested in software development in general, and PSD in particular, and the way in which customer involvement can be achieved in this, the Daydream case was attractive for many reasons. In my research, Daydream represents a software firm focusing on a type of packaged software that must be up-to-date with technological and societal trends. This does not only imply that software developers must be competent in a particular domain but also willing to acquire new knowledge over time. Therefore, interaction with software customers in a virtual customer community can be seen as one source for acquiring such new knowledge. Second, the development of Clusterball involved a commitment to improve its software products and processes by means of a virtual customer community — the Clusterball community. This fact coincided well with my intents to study the role of virtual customer communities for improving PSD. In the Daydream case, the phenomenon of study, I.E. *the role of virtual communities for involving distributed customers in* PSD, was engrained into the context, since the idea with the Clusterball community was precisely this. Thus, in terms of phenomenon of study, the Daydream case offered me an empirical context in which my research interest was clearly discernible. Third, there was the critical aspect of getting access to the phenomenon of study, I.E. the organization, the individuals and the particular system or application of interest. Here, my supervisor played an important role in introducing me to the company after initial contacts had been established. In starting the Center for Digital Business at Umeå University, my supervisor had been in contact with Daydream several times, and in setting up this joint project, Daydream signed as the first collaboration partner in the newly started research center. During the entire time for the study, it was important for me to spend as much time as possible at the research site in order to learn about the company and the way in which it had developed over time. In this, Daydream proved helpful in many ways, providing me with company reports, archived company protocols and development specifications on early versions of Clusterball. In all, reading these documents and spending considerable time at Daydream, added to my understanding of the empirical setting and the larger context of which it was part.

In order to further provide the reader with a good understanding of the empirical setting, the particularities of Clusterball is outlined below (section 3.2.3), as well as the characteristics of the Clusterball community (section 3.2.4). My intention is that these sections will provide the reader with insight in the particular software product that I have studied and the way in which the virtual community was designed to make possible for customer-developer interaction.

3.2.3 *Clusterball*

Clusterball is a multiplayer computer game in which players fly around in ships trying to collect balls and steal balls from other players (FIGURE 1). To protect players from opponents, different kinds of protective equipment can be found in the different venues in which the game takes place. The game uses the 3DGM graphical engine, it is programmed in C++ and it is modeled in Java. As the software for version control, Sourcesafe was used throughout the software development project. In being a sports game — aimed at fascinating people — certain features, i.e., graphics, sound and dynamics, were handled carefully during the development process. First, in terms of *graphics*, Clusterball uses advanced « level of detail » (LOD) technologies. The landscape is rendered with a LOD algorithm and the landing tracks are based on curves that are computed in real time. In this way, Clusterball can be adapted to work on low-end machines as well as using the power of the latest 3D graphic cards. Second, the A3D *sound* API is fully utilized to create a stunning audio environment. Many sounds are physically modeled and directly linked to the dynamics engine to create, for example, collision sounds as realistic as possible. Finally, Clusterball has got sophisticated collision detection and *dynamics* engines. The collision detection uses hierarchical trees of bounding volumes in combination with a fast backtracking algorithm, which grants for fast collision detection that is stable also for low frame rates. The dynamics engine includes a numerical integrator and a full simulation of dynamic rigid bodies including collision response and friction. Also, the aerodynamics of the aircraft is realistically simulated and the users are able to trim the parameters of the aero dynamical model to make the control of the aircraft as intuitive as possible.

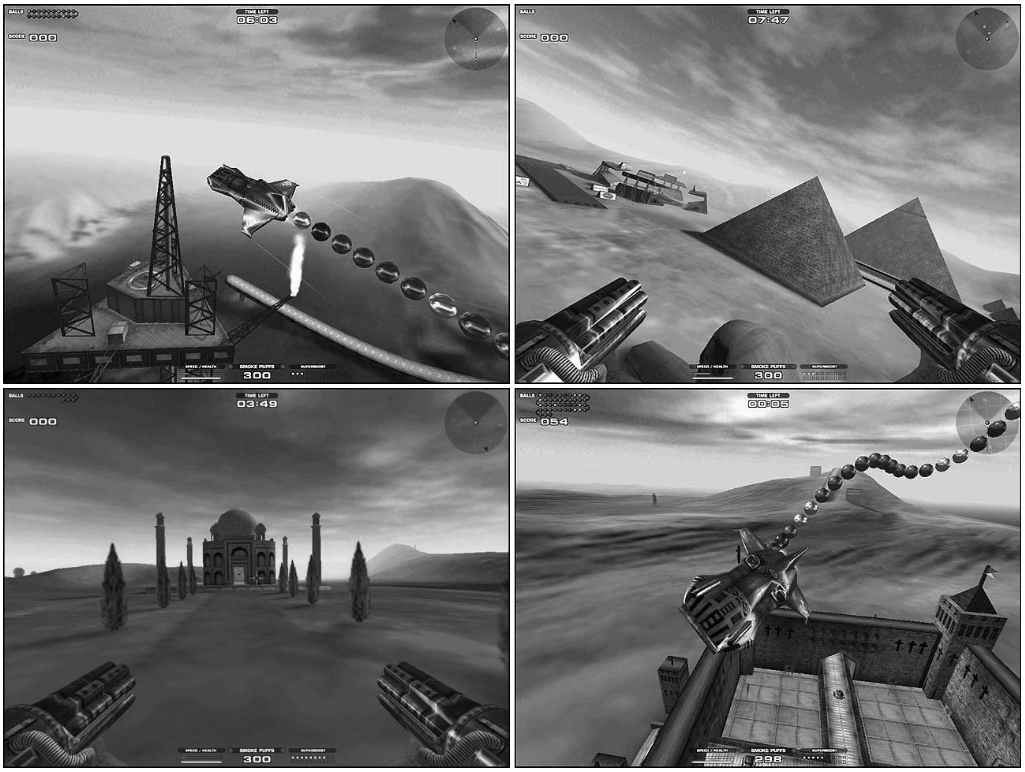
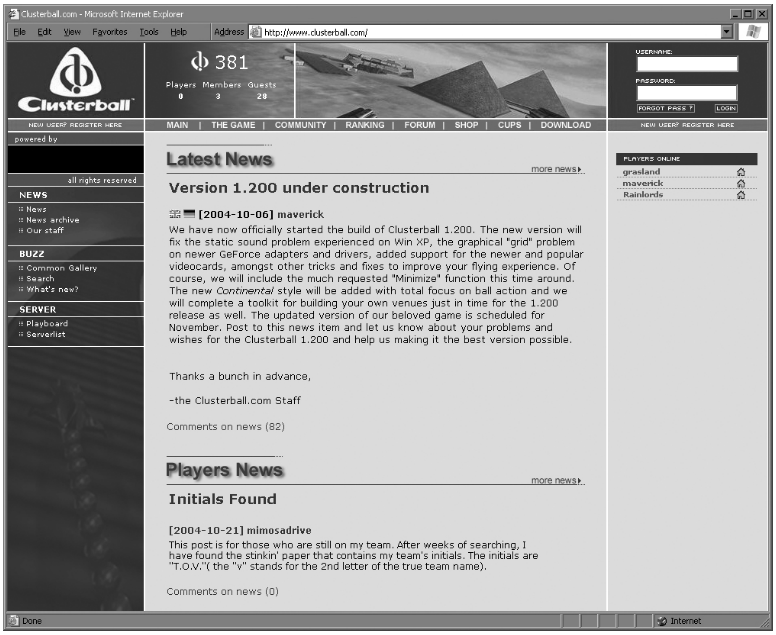


FIGURE 1. Screenshots from four Clusterball venues — Antarctica, Egypt, Taj Mahal and Stonehenge

While there was considerable development work in terms of code generation, there were other challenges related to the idea of Clusterball as not only a packaged software product, but as an *online* game. For example, Daydream introduced an in-house developed network API, called Autobahn. Autobahn uses algorithms for packet compression, packet aggregation and latency hiding which make bandwidth requirements reduced and the system tolerant for packet loss. Due to this, it is possible to set up a Clusterball server on a 56KBPS modem and host 7 other players. The clients need only a 9.6KBPS modem to play. The latency hiding algorithms consist of a force-based system, which is directly linked to the dynamics engine. Without problem, players can compete across the Atlantic, as gameplay is not noticeably affected when network ping is less than 600-700 MS.

3.2.4 The Clusterball community

The Clusterball community, which can be found at WWW.CLUSTERBALL.COM (FIGURE 2), is a game community consisting mostly of members from northern Europe and the US. Depending upon previous game scores, each member is categorized according to the official Clusterball ranking, ranging from « newbie », « bellboy » and « trainee » to « master », « grand master » and « cluster king ». In total, there are 20 different ranking categories and members with the highest rankings are well-known and celebrated members in the community. Together, they engage in discussions concerning Clusterball, and on a regular basis they arrange tournaments and team-play as well as tutorials and training sessions for Clusterball beginners.



Latest News

Version 1.200 under construction

[2004-10-06] maverick

We have now officially started the build of Clusterball 1.200. The new version will fix the static sound problem experienced on Win XP, the graphical "grid" problem on newer GeForce adapters and drivers, added support for the newer and popular videocards, amongst other tricks and fixes to improve your flying experience. Of course, we will include the much requested "Minimize" function this time around. The new *Continental* style will be added with total focus on ball action and we will complete a toolkit for building your own venues just in time for the 1.200 release as well. The updated version of our beloved game is scheduled for November. Post to this news item and let us know about your problems and wishes for the Clusterball 1.200 and help us making it the best version possible.

Thanks a bunch in advance,

-the Clusterball.com Staff

Comments on news (82)

Players News

Initials Found

[2004-10-21] mimosadrive

This post is for those who are still on my team. After weeks of searching, I have found the stinkin' paper that contains my team's initials. The initials are "T.O.V."(the "v" stands for the 2nd letter of the true team name).

Comments on news (0)

FIGURE 2. The Clusterball website (WWW.CLUSTERBALL.COM) where the Clusterball community meet

With approximately 17,000 postings distributed among two different forum tracks (the « general track » and the « technical track ») over a three-year period, the Clusterball community provides an active discussion forum for development and modification of the game. However, CLUSTERBALL.COM is not the only place where the Clusterball community meets. Besides this forum, there are fan websites, I.E. websites developed by community members, that offer forums and chat rooms for community members, and team websites where different Clusterball teams meet and sign up for tournaments. One of the most impressive fan websites is BALLSNATCHERS.COM which was originally developed exclusively for Clusterball by two of the players, and which is now maintained and further developed by a team of Clusterball players from all over the world. Here, the players have their own « hall of fame » (player/team victory announcements), a « haiku corner » (player poems) and a « player gallery » (player portraits).

The common interest in the Clusterball community is computer games in general, and Clusterball in particular. In different forums, community members discuss configuration and installation problems as well as tournaments, team-play and how to improve the game in terms of new functionality. At CLUSTERBALL.COM there is the « technical » and the « general » forum, and at BALLSNATCHERS.COM there is a specific forum for beginners called « young wings » where new players can post questions to the rest of the community. Also, there is a « chat-and-gossip » forum in which the players discuss anything that comes to mind. The devotion and motivation among community members can also be seen in the activities they organize. For example, there are several Clusterball Schools for beginners (see E.G., Ootpek's Clusterschool, Kronix Tips and Lava-Lava's Clusterball Tips at WWW.CLUSTERBALL.COM), a Skin Tutorial in relation to a skin site on which players upload their individually designed skins so that others can download and use them, and a « testimonial site » where Clusterball players share experiences regarding their initial contact with Clusterball.

To communicate, Clusterball community members send postings to electronic forums consisting of different tracks. In these, headings are shown for all topics, and postings are presented as threaded lists. Also, there is a pre-game chat in which players can meet before the match, as well as after, to discuss issues concerning that particular session. In addition to this there are fan websites where several other forums and chats can be found and where many Clusterball players spend time on a regular basis. To manage the community, Daydream created a new position in the company, that of a « community manager ». This person was responsible for responding to the ideas put forward by the community members. According to Daydream, this helped to ensure that the community was nurtured, and that valuable feedback was not lost. Also, many of the developers at Daydream are active community members. Not surprisingly, the Daydream developers can be found in the upper categories on the ranking list since their profound knowledge about Clusterball makes them very skilled players.

3.2.5 My role as a researcher at Daydream

As recognized by Walsham (1995), interpretive researchers are attempting the difficult task of accessing other people's interpretations, feeding them through their own conceptual apparatus, and feeding a version of events back to others. In this, it is important for researchers to have a view of their own role in this process. Either, the role of the *outside observer* or the role of the *involved researcher* can be adopted (Walsham, 1995). As an outside observer, the researcher is seen as not having a direct personal stake in various interpretations and outcomes. The merit of this is that the people studied will hopefully be relatively frank in expressing their views. Of course, the disadvantage is that of not being present on many occasions and thus, not getting a direct sense of the field organization. As an involved researcher, the researcher is to be seen as a member of the field either by participant observations or by action research. The merit of this is the opportunity to get an « inside view » of the field and becoming, at least for a limited period of time, a member of this field. On the other hand, an involved researcher will always be regarded as having personal stake in views and activities, and other personnel may be more careful when expressing their interpretations and opinions. In addition, there is the problem of being an insider but still never « a real insider ». As recognized by Mumford (1985), unless participant observers or action researchers hide their research motives, which could be considered unethical, they will still not be regarded as normal employees and thus, not total insiders.

Despite the difficulties as identified by Mumford (1985), the choice in this study was to adopt the role of the involved researcher. As such, I spent two to three days every week at the research site during a period of six months. As a participant observer I took part in company meetings and workshops, I was part of the project group in developing the Clusterball website and the Clusterball community website, I put together a document evaluating the different Daydream websites and the current status and use of these at that particular time and I took part in the every-day particularities of Daydream. While I suspect that my presence did influence, and sometimes maybe appeared confusing to Daydream employees, there was also the feeling of openness and trust due to the fact that I was there on a regular basis and part of the on-going project group. In taking the responsibility as required as a project member there was always the advantage of getting access to information and also the advantage of being seen as part of the team.

3.3 RESEARCH PROCESS

In all, the research reported on in this thesis reflects a process (2000-2004) during which I have tried my best to explore the research phenomenon as outlined in the introduction (section 1). During these years, my empirical work consisted of a 17 month long study (January 2000-May 2001) at Daydream after which also a follow-up study was conducted (June and October 2002). While the initial contact with Daydream was taken by my supervisor already in December 1999, the research project officially started in January 2000. In accordance with the different types of work that was conducted, the project can be divided into four different phases — *(1) an exploratory study, (2) an in-depth study, (3) a complementary data collection phase, and (4) a follow-up study*. These phases and their different activities, as well as the data material and the research goals of each phase, is summarized below (TABLE 2). Following this summary, each phase is further described in order to provide a detailed account of my empirical work.

Research phase	*Research activities*	*Data material*	*Research goals*
January—March 2000 *Exploratory study*	Workshops at Daydream Formal/informal meetings E-mail correspondence Document review Review of gaming websites	Workshop presentations Meeting minutes Technical/design documents Press releases Shareholders' prospects E -mails Website data	Get an initial understanding of Daydream as an internetworked organization, and the larger gaming context of which it is part.
April—September 2000 *In-depth study*	Participant observations Project meetings Business presentations Product demonstrations Informal meetings E -mail correspondence Website reviewing Development of Clusterball website	Observational data (personal notes) Meeting minutes Presentation material Demonstration material E -mails Website data Community postings Technical/design documents Press releases Patch specifications	Get an understanding of Daydream's software development process and the extent to which customers are involved in this through the use of a virtual community. Get access to interpretations held and enacted by employees at Daydream by being a temporary member of the field.
October 2000—May 2001 *Complementary data collection*	E -mail, telephone and ICQ-correspondence Interview study (Daydream employees) Web survey (community members)	E -mails and ICQ-logs Transcribed interviews Survey forms Community postings Patch specifications	Reveal different perspectives on customer involvement in Daydream's software development process.
June—October 2002 *Follow-up study*	Interview study (Daydream employees) Informal meetings E -mail and telephone correspondence	Transcribed interviews E -mails	Evaluate the role of the virtual community for customer involvement in Daydream's PSD process.

TABLE 2. Summary of the research phases, the research activities, the data material and the research goals in the Daydream study

As can be seen in TABLE 2, the empirical work was initiated with an *exploratory study* was conducted between January and March 2000. In this, we[2] aimed at getting an initial

[2] The Daydream study was conducted by me and a research colleague — Annakarin Nyberg — in a joint project funded by the Center for Digital Business at Umeå University and Daydream Software. As can be seen in this thesis, my research interest focused on the virtual community and the way in which this was used to involve distributed customers in the PSD process. For Annakarin, on the other hand, there was the focus on digital customer relations and how technologies such as the customer relationship management (CRM) database was

understanding of the company and the context of which it was part. During this period we attended company meetings and discussions, reviewed early documents such as design documents and technical specifications of Clusterball and spent time observing the work practice of Daydream employees in order to get a comprehension of the setting, the culture and the study topic (Morse, 1994). During this time, we also presented our research interests and the way in which Daydream provided a relevant empirical setting for our work. Finally, time was spent on reviewing other gaming websites in order to get an understanding of the global gaming community which proved to be a complex network of players, websites and virtual forums. This exploration included analysis of data sources such as technical documents, design documents, meeting minutes and press releases as well as website data such as printouts from different forums and articles published on gaming websites. These activities were aimed at coming to an understanding of the social context and the cultural background of the empirical setting. In this phase of my research, literature on internetworked organizations (Orlikowski, 1999) worked as a source of inspiration, and as a result, I focused on seeing how network technologies made possible for Daydream to involve its customers in the value-adding process of computer game development.

Following the exploratory study, an *in-depth study* was conducted. During this phase, which lasted between April and September 2000, we spent two or three days every week at the research site and a total of 600 hours of participant observations were carried out in order to complement the exploratory study with an in-depth understanding of the development process of Clusterball. In being present at Daydream on a regular basis — in everyday activities ranging from breakfast at the office and informal conversations in the relaxing area to formal project meetings and official business presentations and demonstrations — and in taking active part in the development of the Clusterball website, we had the opportunity to learn about relationships between people, organizations and technology, and also to see how these were not static but constantly changing (Klein and Myers, 1999). During this phase, the aim was to understand the computer game industry as well as the particular challenges that Daydream encountered in being part of this. Also, we came in close contact with both marketers and software developers and learnt that there were many, sometimes conflicting, ideas on how the game should be marketed, and supported by the website. By being active participants in the development of the website we also had the opportunity to be a natural part of the workforce at Daydream. While data from the exploratory phase consisted mostly of already published material, data from the in-depth phase consisted of our own personal documentation in terms of conversation documentation (personal notes and meeting

---

used to keep track of, and learn about, customers and customer behavior. After the first three phases of the study, I continued the Daydream study while Annakarin was involved in another project. For publications related to Annakarin's work, see E.G.:

Nyberg, A., and Henfridsson, O. (2001). Learning about the online customer — an interpretive case study of building digital customer relations in online entertainment. In the *Proceedings of* I3E — *The first* IFIP *conference on* E-*Commerce*, E-*Business*, E-*Government*, B. Schmid, K. Stanoevska-Slabeva, V. Tschammer (EDS.), Zurich, Switzerland, PP. 247-259.

Nyberg, A. and Henfridsson, O. (2001). Going for the Online Customer — An Interpretive Case Study of Internetworked Customer Reach in Online Entertainment. In the *Proceedings of* ECIS *2001*. Smithson, J. Gricar, M. Podlogar and S. Avgerinou (EDS.). Kranj, Slovenia, Moderna Organizacija, PP. 330-338.

minutes) and observation documentation (personal notes). In addition, printouts from the Daydream website, printouts from the community forum and material from official business presentations and demonstrations helped us in following the current debate and the burning topics at the company and among its customers. Entering the in-depth study, my understanding of Daydream as an internetworked organization made me focus on network technologies and the role of these for involving customers in value-adding processes. However, my initial understanding of such processes as general in character was slowly replaced with an understanding of software development processes as specific in character. Hence, while literature on internetworked organization had been important as a starting-point, this was soon complemented with literature on software development and the specifics of software development processes. As a result, research reported from this period reveals customer involvement, and the challenges associated with this, as experienced specifically within software development processes. As part of the in-depth study there was the official release of Clusterball on July 17. At 1.00 pm that day, the first version of Clusterball could be downloaded from the Internet. In being the moment that everybody had been waiting for, the release brought with it both joy and fear. At the company office people were waiting for the first reactions in the community and the preparations for handling initial technical problems had already begun. As expected, there were substantial changes that had to be made and as a result of this, the first software patch was released on July 18, only one day after the official release. Following this, there was the release of the second patch on August 25. In handling technical problems, such as start-up and configuration problems, these patches could be seen as direct responses to the many community postings dealing with this. For me, these patches constituted important data for tracing the way in which customer suggestions, as posted to the community forum, were reflected in Daydream's software development process.

As a rounding-off on our active presence at Daydream, a *complementary data collection* was conducted between October 2000 and May 2001. In this, we maintained close contact to our informants at Daydream. These contacts were upheld by E-mail correspondence, ICQ interactions, and telephone conversations. We also conducted 11 qualitative *interviews* (TABLE 3). In these interviews, our intention was to reveal different perspectives on customer involvement in the development process of Clusterball. Hence, managers, marketing people and developers[3] were interviewed. Each interview lasted for 1-1.5 hours and were both recorded and transcribed. All interviews were of an open character, I.E. we did not direct the interview too closely but instead allowed for the respondents to express their own views in order to gain as much richness as possible for further interpretation (Walsham, 1995).

In addition to the interviews, the complementary data collection phase consisted of a *web-based survey* that was sent out to 200 Clusterball players ranging from « Newbies », I.E. not very experienced players, to « Grand Masters », I.E. very experienced players. The survey,

[3] Due to travels and heavy workload for two Daydream employees, two of these interviews had to be conducted already in August and September 2000 prior to the complementary data collection phase.

consisting of 53 questions (of which 25 were multiple choice questions) was distributed in October/November 2000 and the final answers were collected in February 2001. To be able to represent the wide range of players and their different experiences of Clusterball, the community manager helped us in distributing the survey. In this, we used the customer database to select players from all different ranking categories that had been playing the game at least once the month before the survey, I.E. in September/October 2000. With a response rate of 52 percent, the survey helped us in understanding the players and their apprehension and application of the virtual community. During this phase, there was also the release of four Clusterball patches (October 19, December 20, February 22 and April 29). In handling bugs and errors as well as in including several new features such as replay and recording, pre-game chat, the possibility for match-making and ranking ETC., these patches attracted much attention in the community and in the community forum this was evident in a rising number of postings.

Title	*Date*	*Title*	*Date*
Marketing manager	2000-08-26	Manager/chair of board	2000-11-02
Executive director/founder	2000-09-21	Developer/founder	2000-11-14
Marketer	2000-10-26	Developer #2	2000-11-28
Developer #1	2000-10-26	Community manager #1	2000-11-28
Administrator/external relations	2000-11-02	Community manager #2	2001-05-08
Administrator	2000-11-02		

TABLE 3. Interviews conducted during the complementary data collection phase

For me, the first three phases of the Daydream study (*the exploratory phase, the in-depth study and the complementary data collection phase*) provided me with a rich understanding of the empirical context. As a point of departure, there was my inherent interest in *customer involvement* and the way in which this could be achieved in development processes. In the exploratory as well as in the in-depth study, this interest was discernible in relation to value-adding processes within organizations using network technologies for interacting with customers. Here, Orlikowski's (1999) notion of *internetworked organizations* was used as a theoretical lens through which Daydream's development process was interpreted and understood (PAPER 1), and as a result, the virtual community was portrayed as a mediating technology characterized by openness and accessibility. In the later stage of the in-depth study, however, I started to see the particularities of Daydream's development process as critical. While the notion of internetworked organizations could apply to any organization intending to use network technologies for mediating purposes such as customer involvement, I soon realized that the way in which this mediation was carried out at Daydream was closely dependant on the characteristics of their process, I.E. the particularities of the software development process. As a result of this, literature on *software development* came to work as the foundation for further interpretation of my empirical findings (PAPER 2 and PAPER 3). As a common characteristic, the papers reflecting these early research phases very carefully report on the empirical setting of Daydream and the particularities it embraced in terms of technology, business models and software development practices.

After the complementary data collection phase ending in May 2001, our Daydream study was rounded off. At that time we had spent 17 month working closely with people at Daydream and the opportunity of being part of development, release and further improvement of the product had given us a good understanding of the organization and the way in which Daydream's software development processes were conducted. However, while my research colleague at that time chose to direct her focus to another study, I chose to conduct an additional *follow-up study* with some of the people that had been interviewed during the complementary data collection phase. The aim of this study was to further strengthen my data collection and to be able to evaluate the role of the Clusterball community for customer involvement. This follow-up study was conducted in June and October 2002. In this, I conducted *interviews* with the marketing manager, the lead programmer and the graphical designer (TABLE 4) in order to grasp changes and to reach further understanding of the opportunities and challenges that Daydream had experienced during the development process of Clusterball. Each of these interviews lasted for 1.5 hours and were both recorded and transcribed. In looking back at the development of Clusterball, these interviews were of importance for evaluating the role of the virtual community and the way in which different stakeholders perceived it important for customer involvement.

Title	*Date*
Marketing manager	2002-06-14
Lead programmer	2002-10-24
Graphical designer	2002-10-24

TABLE 4. Interviews conducted during the follow-up study

In providing me with additional data on the role of the community, the follow-up study encouraged me to write more critically on the topic. As discernible in the interviews from this study, there were many benefits of using the community for involving customers in the software development process — but also, challenges were discernible. From having written mostly positive accounts on community use, these interviews made me re-evaluate many of my early findings and put them in perspective according to what this follow-up study revealed in terms of challenges and difficulties. In allowing for a more critical view on community use, the follow-up study made me caution against the tendency to romanticize the community construct and especially that of virtual communities (Schwen and Hara, 2003). While my more critical accounts can be seen in PAPER 5 and PAPER 6 in this thesis, PAPER 4 reflects the changing nature of software development environments which was also a topic that came to me after having left — and having had the time to further reflect upon

— the research site for some time. In this process, I came across literature on *packaged software development* and inspired by this, I started to evaluate my research in accordance with this. While the theoretical understanding acquired during early research phases had been influenced by conceptions of « internetworked organizations » and « software development », the follow-up study made me evaluate and present my research in terms of PSD (Sawyer, 2000, 2001; Carmel and Sawyer, 1998; Keil and carmel, 1995) and the conceptual apparatus as found within the knowing-in-practice literature (see E.G., Boland and Tenkasi, 1995; Tsoukas, 1996; Brown and Duguid, 2001; Orlikowski, 2002). To further describe how my empirical work was intertwined with theoretical conceptions in order to form and refine my understanding for the research phenomenon, this process is elaborated upon below.

3.3.1 The research process as a transition between empirical data and theoretical concepts

Guided by the hermeneutic principle, my research process started with a research interest of an open character (Patton, 2001). Rather than having a focused research question, my initial formulation of the problem domain was very broad, intended not to constrain coming analysis but to make possible for different perspectives to be adopted. As is common in long-term research, the initial conceptual apparatus encompassing certain assumptions and beliefs, transformed over time and as suggested by the hermeneutic principle (Klein and Myers, 1999; Patton, 2001), transitions between theoretical conceptions and empirical data were central for me in articulating an increasingly plausible understanding of the research phenomenon outlined in this thesis. Throughout my research process, this meant that my early understanding of customer involvement was continuously refined through transitions between theoretical conceptions and empirical data.

Theoretically, knowing-in-practice literature (see E.G., Boland and Tenkasi, 1995; Tsoukas, 1996; Brown and Duguid, 2001; Orlikowski, 2002) provided me with an apparatus for understanding concepts such as « knowledge », « practice » and « community ». For me, situated customer knowledge was of primary interest, and in conceptualizing this as a capability enacted in practice, where practice is defined as the situated, recurrent activities of human agents, this literature successfully helped me in articulating the inherent complexity and multiplicity of the phenomenon I sought to portray. As a result of my interpretive approach in which understanding is something that is emerging throughout the research process, a major part of my theoretical understanding is discernible in the later phases of the research process and hence, reflected mainly in the final papers of this thesis. While I already in the beginning was influenced by the theoretical conceptions as mentioned above, my deeper understanding and hence, more thoughtful application of these was something that emerged after having had time to consciously reflect on these and iterate between these concepts and my empirical data.

Empirically, the Daydream context embraced ingredients important for me and my research interests. Through interviews and participant observations I gained insight in both customers' and developers' apprehension of the development process of Clusterball. With the possibility to discuss these with the informants as well as with my research colleague, my understanding of the context was early articulated and continuously refined. Intertwined with theoretical conceptions, my interventions in the everyday practices of Daydream and my close interaction with Daydream employees made me see that what my research would describe was my own interpretation of this context, not something that was absolutely correct or true (Patton, 2001). In engaging in this context on a daily basis, an understanding of its particularities soon emerged and I begun to realize that what I was doing was constructing a reality on the basis of my empirical experiences (Patton, 2001) and theoretical insights. In this way, my emerging understanding of this context can be seen as "*...an infinite process, ending only when a sensible meaning and a coherent understanding has been reached*" (Kvale, 1987, s. 62). For me, this was indeed an infinite process, reaching far beyond this thesis. While each individual paper articulates parts of this understanding, the outcome in terms of "*...sensible meaning and a coherent understanding*" (Kvale, 1987), is reached only in relating these parts to a larger context. In terms of the papers in this thesis, and in similar with the process of interpretation (Klein and Myers, 1999), the first papers can be seen as reflecting a precursory understanding for the final outcome, I.E. the different empirical *parts*, while the latter papers can be seen as reflecting the context for the outcome, I.E. the *whole* as a result of my transitions between empirical parts and theoretical concepts. In this way, I hope to enable the reader of this thesis to gain insight in the research phenomenon by contrasting empirical insights with theoretical expressions of the whole. As well as my own understanding of the research phenomenon was achieved through continuous transition between empirical insights and theoretical conceptions, the parts and wholes as expressed in the different papers will enable the reader to reconstruct these transitions and hence, follow me in my process to a conceptual understanding of virtual community use in PSD.

As a result of theoretical and empirical insights, and the continuous interplay and transition between these, this thesis presents a conceptual model for understanding *community use* in PSD (FIGURE 5 in section 4.2.3). In this model, knowledge building, knowledge elicitation and knowledge exploitation are suggested as comprehensive terms denoting my understanding of knowledge creation and transformation processes important for community use in PSD. In reflecting my own conceptions of the research phenomenon, this model, and the knowledge processes identified in this, is my theoretical contribution — primarily analytical as to facilitate an understanding of community in PSD.

3.3.2 Adopting Klein and Myers (1999) principles for the conduct and evaluation of the Daydream study

As introduced in section 3.1.2, there are well-established principles for conducting and evaluating interpretive research. These principles can be seen as a response to the call "*...to discuss explicitly the criteria for judging qualitative research in information systems*" (Lee ET AL, 1995, P. 367), and were proposed by Klein and Myers in a paper published as part of a special issue in the MIS *Quarterly* in 1999. In crystallizing a diffuse literature into a manageable set of principles, Klein and Myers (1999) make possible for better quality assessment of interpretive research as well as for the individual interpretive researcher to design their investigations more carefully and systematically. In the Daydream study, these principles were considered in both research design and research evaluation. Below, a summary of these principles are provided (TABLE 5). Following this, I discuss how each of these principles was accounted for in this particular study.

---

1. *The Fundamental Principle of the Hermeneutic Circle*
This principle suggests that all human understanding is achieved by iterating between considering the interdependent meaning of parts and the whole that they form. This principle of human understanding is fundamental to all the other principles.

2. *The Principle of Contextualization*
Requires critical reflection of the social and historical background of the research setting so that the intended audience can see how the current situation under investigation emerged.

3. *The principle of Interaction Between the Researchers and the Subjects*
Requires critical reflection on how the research materials, or data, were socially constructed through the interaction between the researches and participants.

4. *The Principle of Abstraction and Generalization*
Requires relating the idiographic details revealed by the data interpretation through the application of principles one and two to theoretical general concepts that describe the nature of human understanding and social action.

5. *The Principle of Dialogical Reasoning*
Requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings with subsequent cycles of revision.

6. *The Principle of Multiple Interpretations*
Requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the same sequence of events under study. Similar to multiple accounts even if all tell it as they saw it.

7. *The Principle of Suspicion*
Requires sensitivity to possible « biases » and systematic « distortions » in the narratives collected from the participants.

TABLE 5. Summary of principles for conducting and evaluating interpretive field research (Klein and Myers 1999)

In taking into account the *principle of contextualization* (principle 2), the Daydream study was designed with an initial exploratory phase in which we aimed at getting an understanding for the social and historical background of Daydream. In this phase, company documents and product specifications were reviewed, as well as global gaming websites, in order to get a picture of the empirical context of which Daydream was part. Furthermore, the principle is discernible in the writing of research papers (PAPER 1,PAPER 2 and PAPER 3). In focusing on the empirical setting and the particularities associated with this, these papers reveal the background to the Clusterball development project, I.E. initial Daydream visions, as well as changes and modifications to these. In reading these papers, the reader will understand the emergence of the Clusterball project, the way in which it developed and the challenges that Daydream encountered during the project.

Similarly to the *principle of contextualization* (principle 2), *the principle of interaction between the researchers and the subjects* (principle 3), *the principle of dialogical reasoning* (principle 5) and *the principle of multiple interpretations* (principle 6) were all considered in research design and in the published papers. In relation to possible contradictions, differences in interpretations and how data can be socially constructed, as reflected on in these principles, the study was designed so that we were two researchers present at the research site. While having different research questions and hence, different interests and methods for exploring these, there was the opportunity for us to continuously discuss and reflect upon our presence at the research site, the interpretations that we made and the contradictions we encountered in relation to theoretically founded preconceptions and the story that the data told. In terms of published papers, the above mentioned principles are most evident in the latter papers (PAPER 5 and PAPER 6). Here, there is a critical investigation of the limitations of virtual community use in PSD and it is evident that theoretically grounded forecasts of virtual community success do not always come true. While earlier papers (PAPER 1 and PAPER 2) to a greater extent reveal virtual community success, these latter papers add to the story by bringing in a more critical perspective as advocated also in the *principle of suspicion* (principle 7). In doing this, these papers not only reflect the different principles. As important, they reflect my own research process and the understanding I developed during the time for this study. My early papers (PAPER 1 and PAPER 2) reveal opportunities with virtual community use. As a contrast, the final papers (PAPER 5 and PAPER 6) reveal also disappointments and failures with community use, something that became evident to me after having left the site, providing time to reflect on what my research subjects actually meant in the words and actions that were presented to me. Of course, this enhanced understanding of the field emerged as a result of my own reflection, but in explicitly stating how important such reflection is, the principles as proposed by Klein and Myers (1999) have significantly contributed to this achievement.

Finally, the *principle of abstraction and generalization* was considered in the way that my empirical data was collected and analyzed. Following the interpretive tradition in general and the principle of abstraction and generalization in particular, my data analysis consisted of an iterative process in which I combined empirical data of the development process of Clusterball and the use of the Clusterball community with theoretical concepts of « knowledge », « practice » and « community » (see E.G., Boland and Tenkasi, 1995; Tsoukas, 1996; Brown and Duguid, 2001; Orlikowski, 2002). In my analysis, I used an inductive approach (Patton, 2002) to find patterns, themes and categories to be used for further analysis of the data. As a result of this process, the material was categorized into different categories focusing on different aspects of the case. These categories were of an open character (Strauss and Corbin, 1998), with the purpose to categorize different stakeholders' statements and expressions. The categories that were used were not predefined, but instead they were emerging as a result of me interacting with the data and getting an understanding of the content in relation to the research phenomenon. For example, as a result of this transitional process there is the concept of *community knowledge* as presented in Paper 6. Here, the particulars of the Clusterball community and the way in which it was used, is related to the concept of « knowledge » (see E.G., Boland and Tenkasi, 1995; Brown and Duguid, 2001) and it is suggested that virtual communities constitute platforms for community knowledge important for improving PSD. In this, literature on organizational knowledge proved useful as a theoretical lens through which the particulars of my empirical data would be interpreted.

4 A community-based perspective on PSD

This thesis is based on a conception of software development as a knowledge intensive activity (Clegg ET AL, 1996; 1997) in which important knowledge is residing outside the traditional boundaries of the software firm (Lee and Cole, 2003; Orlikowski, 2002). In accordance with this conception, knowledge creation processes expand beyond what is referred to as « the level of the firm », and, as suggested in this thesis, software development processes would benefit from utilizing also this knowledge. As recognized by Finch (1999) and Von Hippel (1986), many of the best ideas for product improvements come from the customer community. Still, customers remain a largely untapped source for knowledge creation (Nambisan ET AL, 1999) and are not integral to contemporary PSD efforts (Sawyer, 2000). Hence, the ability to incorporate distributed customer knowledge becomes a critical challenge for any organization aiming for further expansion of knowledge or innovation in their product development processes (Lee and Cole, 2003).

Recognizing the importance of knowledge residing outside the boundaries of the firm, I.E. knowledge residing in distributed customer communities, this thesis explores the *role of virtual communities for involving distributed customers* and hence, for involving distributed customer knowledge, into the PSD process. The recognition of customer involvement as beneficial for the PSD process (Keil and Carmel, 1995) as well as the idea of customers as co-creators in product development (Nambisan, 2002) has brought with it the view of new product development as a knowledge intense activity highly dependent on its customers. Referred to as « communities of knowing » (Boland and Tenkasi, 1995), « knowledge nets » (Barnes, 1983) or « communities of practice » (Wenger, 1998; 2000), the idea of communities as enablers for knowledge creation has for a long time been prevalent in the field of IS research. In accordance with many sociological definitions of communities emphasizing features such as relationships and social interaction (see E.G. Hillery, 1955; Wellman and Gulia, 1999; Rheingold, 1994; Baym, 1998), this view portraits communities as dynamic structures in which the creation of knowledge is due to a continuous questioning and revision of routines, processes and relationships among community members (Boland and Tenkasi, 1995). Here, knowledge is nor individual nor communal, but a context-dependant resource (Tsoukas, 1996), experienced and manifested by the engagement and participation of each individual in the practice of which the community is part (Wenger, 1998).

Below, a brief introduction to the concept of communities is given and some of the prominent definitions for describing the phenomenon are outlined (section 4.1). Following this (section 4.2), I discuss the role of communities as important for knowledge creation and knowledge transformation processes. To this end, I present a model for understanding community use in PSD. As a result of my interpretive approach to research in which theoretical and empirical iterations have contributed to my overall understanding of the research phenomenon, the *community use model* represents my understanding of the conditions under which community knowledge may be built, elicited and exploited for the purpose of PSD improvement. Hence, the community use model depicts my understanding of how a community-based perspective may be an approach for involving customers in PSD.

4.1 COMMUNITY: BACKGROUND AND DEFINITIONS

There is little doubt that the advent of the Internet and Web-based technologies has enabled communities of geographically distributed people to convene and interact in ways not experienced before. As a powerful tool for cheap, fast and global interaction, the Internet brings with it the opportunity for millions of people to create social spaces where they can organize themselves and share resources via electronic interfaces (Smith and Kollock, 1999). The concepts of « communities » and « belonging » have been transferred to the virtual world, and today we regard virtual interaction and virtual relationships as « real » in the sense that they complement, and sometimes even replace, physical interaction and face-to-face meetings between people.

Still, ambiguity exists concerning how to define « community ». As pointed out by Fernback (1997, P. 39) this is due to the fact that the term "*...has descriptive, normative and ideological connotations...and encompasses both material and symbolic dimensions*". To further complicate, there is the recognition of the many perspectives from which communities can be studied (Preece, 2001, see also TABLE 6). First, the *sociology perspective* emphasizes features such as size, location, relationships and social interaction (see E.G., Hillery, 1955; Wellman, 1997; Rheingold, 1994; Baym, 1998; Hamman, 2001). Second, the *technology perspective* emphasizes technical infrastructure, architecture and the design of the community-supporting software (see E.G., Stanoevska-Slabeva and Schmid, 2001; Lechner and Schmid, 2001). Third, the *virtual world perspective* emphasizes the sense of immersion that mimics reality and prolonged, repetitive interaction such as the interaction found in E.G., MMOs and MUDs (see E.G., Curtis, 1992; Dibbel, 1998; Pargman, 2000). Finally, the *E-commerce perspective* emphasizes communication and information transfer, marketing purposes and the opportunity to use community features to draw people to a specific website (see E.G., Hagel and Armstrong, 1997; Kim, 2000).

In an attempt to present one clear definition of « community », Hamman (2001) uses an analysis made by Hillery (1955). Here, 94 sociological definitions of the term were subjected to qualitative and quantitative analysis in order to identify concepts that were common in the sample of different definitions. In his study, Hillary found only one concept that was common among the 94 definitions — they all dealt with *people*. However, there were other areas where the majority of the different studies were in agreement. As a result, Hamman (2001, P. 75) suggests that the sociological term « community » should be understood as meaning (1) a group of *people*, (2) who share *social interaction*, (3) and some *common ties* between themselves and other members of the group, and (4) who *share an area* for at least some of the time. Other prominent definitions are those presented by Rheingold (1994) and Whittaker ET AL (1997, P. 137). In these, « personal relationship », « companionship », « shared goal » and « shared context » are identified as core attributes of a community. In his commonly quoted definition, Rheingold (1994, P. 5) describes virtual communities as:*"...social aggregations that emerge from the Net when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace"*.

Community perspective	*Description*	*Research*
Sociology perspective	Definitions emphasizing features such as size and location, and more recent definitions emphasizing relationships and social interaction.	Hillery (1955) Wellman (1982) Rheingold (1994) Baym (1998) Hamman (2001)
Technology perspective	Definitions emphasizing technical structure and architecture, technical functions and the design of the community-supporting software.	Stanoevska-Slabeva and Schmid (2001) Lechner and Schmid (2001)
Virtual worlds perspective	Definitions emphasizing a sense of immersion that mimics reality and prolonged, repetitive interaction such as the interaction found in E.G., MMOS and MUDS.	Curtis (1992) Dibbel (1998) Pargman (2000)
E-commerce perspective	Definitions emphasizing communication and information transfer, marketing purposes and the opportunity to use community features to draw people to a specific website, E.G., stickiness.	Hagel and Armstrong (1997) Kim (2000)

TABLE 2. Examples of different perspectives from which communities can be studied

In the field of information systems (IS) the term « community » is transferred to describe the nature of computer-mediated interaction that goes beyond the mere function of communication — for mediating the negotiation (Wenger, 1998) and transformation of relationships (Boland and Tenkasi, 1995) that takes place among community members. Here, the medium itself becomes important in bringing geographically distributed people together, and in addition to the social aspects of community, the definitions include also technical aspects. For example, Preece (2000) defines virtual communities in terms of « people », « purpose », « policies » and « computer systems ». Similarly, the definition provided by Stanoevska-Slabeva and Schmidt (2000) introduces « technological mediation », « ubiquity » and « online identity » as distinguishing features imposed on communities by the usage of the digital medium. Mynatt ET AL (1997) introduces the concept of « network communities » and argues that these embody a unique constellation of characteristics that distinguish them from other forms of media and from other types of computational systems. Finally, recent research on system architecture and community infrastructure suggests that virtual communities should be described both in terms of its members and in terms of the platform, and that the technical aspects are significant for the overall community building process (Hummel and Lechner, 2001; Stanoevska-Slabeva and Schmidt, 2001).

In looking at these different definitions, it should be noted that the term « virtual » has added another dimension to consider when defining « community ». In contrast to sociological definitions in which place and physical presence are central aspects, it is clear that computer networks allow for communities that stretch well beyond the neighborhood (Wellman and Gulia, 1999). Below (FIGURE 3), and in an attempt to summarize the different notions of « community », the four characteristics as presented by Hamman (2001) are used to portray communities in terms of their *social aspects*. In addition to this, *technical aspects* (Sanoevska-Slabeva and Schmidt, 2001; Preece, 2000) are added to also portray communities as technical infrastructures enabling virtual communities to expand physical or organizational boundaries. For this purpose, synchronous and asynchronous technology such as websites, E-mail systems, electronic discussion forums and chats are used (Preece, 2000). In understanding communities in terms of both social and technical aspects there is the possibility to view them as enablers for knowledge creation among geographically

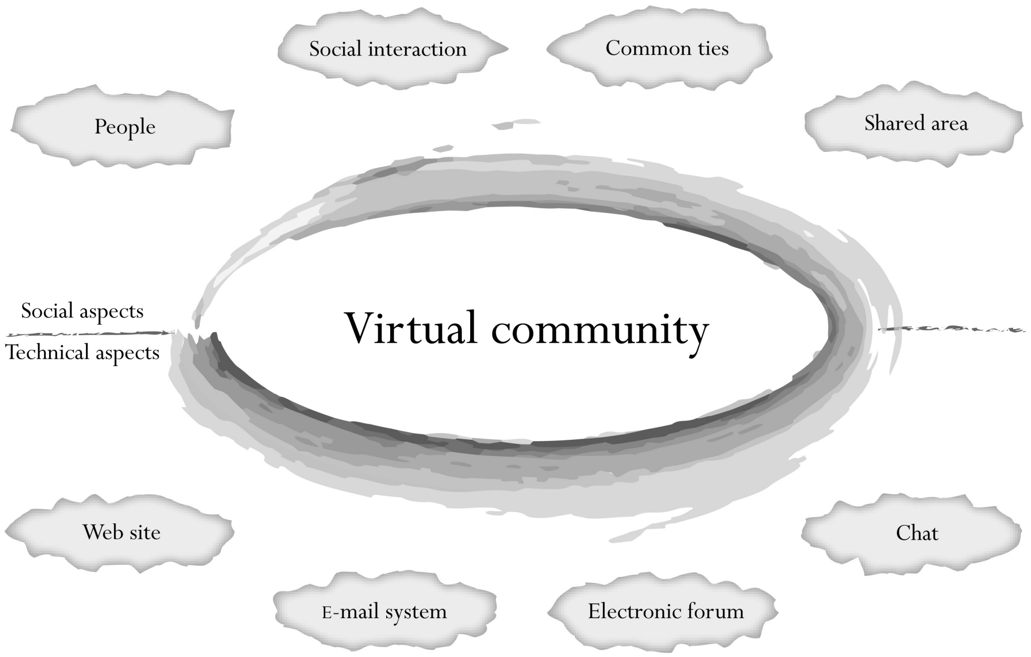


FIGURE 1. A framework for understanding virtual communities in terms of social and technical aspects

distributed people, in this case among distributed software customers. While the characteristics in terms of people, social interaction, common ties and shared area are all necessary for communities to develop and maintain, there is the additional need for an infrastructure in terms of technology for virtual communities to emerge and sustain. In this, virtual communities exist at the intersection of social and technical systems (Stanoevska-Slabeva and Schmidt, 2001), and as recognized by Mynatt ET AL (1997), neither technology nor sociality can supplant the need for the other, and the two are conceptually inseparable. Consequently, there are two constitutional elements of virtual communities — the association of community members (as represented by the « social aspects » in FIGURE 3) and the enabling electronic medium (as represented by the « technical aspects » in FIGURE 3).

In terms of community knowledge, this is dispersed among geographically distributed community members and as recognized by Lee and Cole (2003) community knowledge creation is no longer abundant within, for example, a firm or a specific group of people, but instead dispersed across geographical and organizational boundaries. In using the development process of the Linux kernel as an example of community-based knowledge creation, Lee and Cole (2003) provide an interesting example of how knowledge residing among distributed Linux users is made explicit in the virtual community and how exploitation of this knowledge benefits the open source software (OSS) development process. While OSS communities may act as an illustrative example of community-based knowledge creation, examples can be found within traditional software firms as well.While not being able to access the source code of the software, and therefore not being part of code generation, these communities focus on discussing, testing and providing suggestions for the improvement of different software products. In this way, they constitute an important resource in the process of software development and improvement. Below, the idea of virtual software communities is further outlined to reflect the opportunity of using these for customer involvement in PSD.

Virtual software communities
As one example of successful software communities there are the OSS communities. Here, distributed software users meet to share information and collectively develop software applications, predominantly operating systems and web application systems (Feller and Fitzgerald, 2002). Membership in these communities is voluntarily and, instead of monetary compensation, contribution is based on motivation. What the community provides is a platform for collaboration and exchange of information between software users. Several case studies highlight the particularities and the fascination of open source (Mockus ET AL, 2002; Moon and Sproull, 2000) and for example, motivational factors and trust issues (Gallivan, 2001), distributed knowledge creation processes (Lee and Cole, 2003) and the many pros and cons of OSS development (Jørgensen, 2001; Sharma ET AL, 2001) have been studied in order to see to what extent the open source paradigm can be transferred to traditional organizing and traditional software development. As recognized by Ljungberg (2000), there are reasons to believe that OSS development has the potential to influence the future of organizations both in terms of organization, customer relations and business models.

Besides OSS communities there are other software communities in which users of software play an important role. *Virtual customer communities* have become a dominant feature in association to software product websites on the Internet. While not being able to access the source code of the software and therefore not being part of code generation, these communities focus on discussing, testing and providing suggestions for future software improvements. Today, the creation of virtual customer communities has captured popular, as well as scholarly attention. Not surprisingly, new business opportunities has emerged and already there are several companies offering software solutions for creating virtual customer communities in which people can collaborate and exchange ideas independent of location and time. For example, there is Web Crossing (WWW.WEBCROSSING.COM) and Ramins (WWW.RAMINS.NET) which provide companies such as NetSuite and Nortel Networks with software for having distributed customers meet virtually to discuss their products. As it seems, these companies believe that knowledge inherent in their customer communities can provide valuable input to their development processes. As acknowledged by IBM, their customer communities provide *"...an open forum for easy exchange of information as well as valuable information to* IBM *development and support"* (WWW-306.IBM.COM, 2004-03-01). In reading this statement, much of the essence of virtual customer communities is captured as well as the context within which they often appear. As can be seen in the IBM example, easy exchange of information is regarded the driving force in a customer community and the members enjoy the opportunity of having access to other peoples' experiences of particular products. Many times, customer communities can be seen as a problem solver in that customers benefit from helping each other and due to mutual engagement (Wenger, 1998) and reciprocity (Smith and Kollock, 1999), information exchange and retrieval is smooth and fast. Regarding the context, virtual customer communities appear in association to development and support of particular software products or software development platforms. For example, there are the CATIA customer communities (WWW.TENLINKS.COM) discussing CAD/CAM products, the IBM Tivoli customer communities (WWW-306.IBM.COM) discussing a particular software product for intellectual capital management and access. In association to development tools and platforms, there are the Java Technology customer communities (DEVELOPER.SUN.COM) in which more than 500 groups in 100 countries discuss Java technology and Sun products and there is the GapiDraw customer community (WWW.GAPIDRAW.COM) in which GapiDraw — a graphics platform for creating applications on handheld computers and Smartphones — is discussed by its international community of customers. In allowing for discussions, for testing and for product suggestions, these communities can be seen as enablers for customer involvement in PSD. Below, a theoretical discussion is provided in order to create an understanding of virtual customer communities for customer knowledge creation and transformation processes important to PSD.

4.2 COMMUNITY USE IN PSD

Viewing software development as a dynamic and knowledge intensive activity (Clegg ET AL, 1996; 1997) brings with it the understanding of customer knowledge, skill and expertise as central for the development process and outcome (Clegg ET AL, 1997). As recognized by Keil and Carmel (1995) and Nambisan ET AL (1999), software development processes benefit from customer involvement. Indeed, many of the best ideas for product improvements often come from customers (Finch, 1999; Von Hippel, 1986). With profound experience and detailed knowledge of specific software products customers can be seen as possessing situated knowledge of the software and the particular situations in which it is used. Hence, customers — and the knowledge they possess — constitute an important resource in the process of software development.

While the understanding of customer knowledge as important to software development is not new, PSD has long suffered from lack of customer involvement. Due to *distant customers* (Sawyer, 2000) and *unknown customers* (Grudin, 1991), packaged software customers are often identified when development ends and the product goes to market (Keil and Carmel, 1995). Also, the physical distance between customers and developers entails logistical and economical problems that add to the challenge of involving customers in the development process (Nambisan, 2002). Instead, intermediaries and customer surrogates have been used and while the question is not *whether* customers should participate in the development process, it is a matter of *how* they should efficiently do this (Sawyer, 2000; Keil and Carmel, 1995). In other words, while customer knowledge is recognized as critical to PSD, existing literature does not deal with the challenge of designing organizational capabilities to cater for this. Below, *community knowledge* is proposed as a comprehensive term for understanding customers' situated knowledge on particular software products (section 4.2.1). In order to build, elicit and exploit such knowledge, I propose the *community knowledge use cycle* as consisting of three iterative and interrelated processes (section 4.2.2). Finally, and as a result of my theoretical and empirical understanding of community knowledge and how such knowledge may be achieved in the context of PSD, I present the *community use model* (section 4.2.3). This model is a conceptual understanding representing external environmental conditions as well as internal knowledge transformation processes important for community-based customer involvement in PSD.

4.2.1 Community knowledge

As recognized by Orlikowski (2002), knowledge inherent in domain-specific customer communities is a type of « knowing-in-practice » that is socially accomplished and constituted and reconstituted in the everyday practices of which those people are part. Furthermore, this knowledge is not stable or enduring but rather a capability that is enacted in each moment of action and that emerges only through ongoing relationships of context, actions and structures. Thus, peoples' interpretation and experience of doing « the same thing » fosters knowledge that can be seen as representative for a group of people, I.E. a community. In Wenger (1998), this is understood as « communities of practice » in which people engage in shared practices, I.E. joint enterprises, and where knowledge is manifested in the mutual engagement in this enterprise and in the shared repertoires of participants. According to Wenger (1998), communities of practice exist everywhere. They are the basic building blocks of any social system because they are *"[...] the social « containers » of the competences that make up such a system"* (Wenger, 2000, P. 229). Consequently, we all belong to communities of practice. By participating in these communities we define with each other what constitutes knowledge in a given situation, and it is argued that participation in communities of practice is *"[...] the very core of what makes us human beings capable of meaningful knowing"* (Wenger, 2000, P. 229). Here, continuous negotiation and re-negotiation is necessary to form what can be referred to as community knowledge, and while this knowledge is often described as intra-organizational (Lave and Wenger, 1991) there is also the recognition of community knowledge as residing outside the boundaries of the firm (Lee and Cole, 2003). Likewise, Boland and Tenkasi (1995) use the label « communities of knowing » when describing the expertise of different knowledge groups both within the firm and between the firm and its surrounding environment.

Wherever located, knowledge of particular communities is situated and associated to the everyday practice of which the community is part. As recognized by Brown and Duguid (2001), what we know always depend on — and reflect — the social context in which this knowledge is acquired and put into practice. In this thesis, this situated knowledge is referred to as *community knowledge* which is a type of knowledge that is highly context-dependent and manifested in the practice of different social settings. Apart from community knowledge as existing within the practice of software firms, community knowledge in the context of PSD should be understood as customers' situated knowledge on particular software products. Hence, it reflects not only technical knowledge such as hardware and software configuration skills, but also the particular circumstances and purposes of software use. As such, community knowledge is enacted in the moment and emerges only through everyday software use.

As noted by Wenger (1998), community knowledge belongs to the community in that it is a capability that emerges through communal negotiation and communal experience, in this case communal negotiation and experience of software use. However, despite belonging to the community, this does not imply that community knowledge cannot be supported and encouraged by organizational activities in order to become useful for the software firm. On the contrary, organizational use of community knowledge is critical in order to transform this knowledge into purposeful action, such as for example IT innovation (Nambisan ET AL, 1999), or as in this case, software improvements. In similar with Wenger's (1998) conception of communities as independent and autonomous, but at the same time impressionable in terms of help and support, this thesis argues for community knowledge as practically relevant only when it is transferred from within the community to its surrounding environment, I.E. the software firm and its' PSD processes. If this transformation is achieved, I.E. if customers' situated knowledge on particular software products is made accessible to the software firm, there is the potential for software improvements reflecting accurate customer needs and requirements. Also, successful transformation of community knowledge increases customer involvement in PSD, which previously has been identified as difficult to attain (Sawyer, 2000).

4.2.2 Community knowledge use

As recognized above, situated knowledge as existing in distributed customer communities, is critical for improving packaged software products and processes. In using this knowledge, software firms seek to benefit from collective idea generation among geographically distant customers (Nambisan, 2002). However, to succeed in this community knowledge use, I.E. in transforming community knowledge into software improvements, there are certain organizational processes that need to be undertaken. In this thesis, I propose a model of community knowledge use in which these processes are understood as three iterative and interrelated processes unfolding at the intersection between commercial software firm practices and voluntary community participation (FIGURE 4).

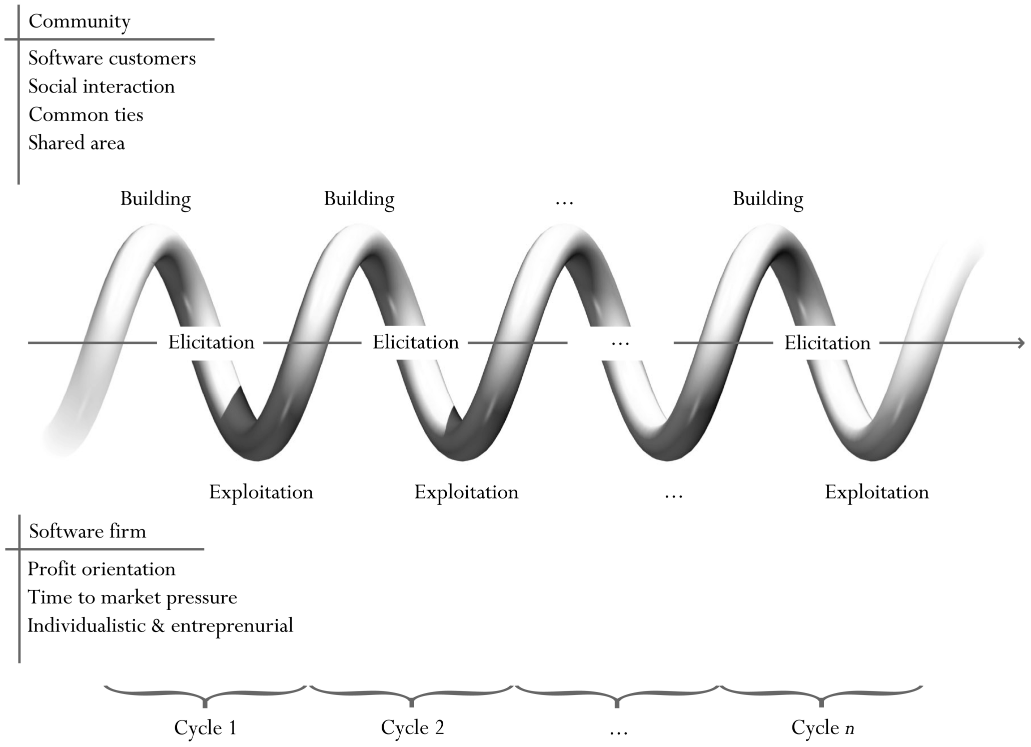


FIGURE 4. The community knowledge use cycle
(Holmström and Henfridsson, submitted)

In my model, a software community is understood as a group of *software customers* sharing an interest in a particular software product (having *common ties*) and using a virtual community for interacting and coalesce around this interest (using virtual community infrastructure as a *shared area* for *social interaction*). In accordance with the understanding of customer knowledge, skill and expertise as central for the development process and outcome (Clegg ET AL, 1997), this model views software communities as resources in providing the software firm with valuable input in terms of, for example, graphic skills, hardware and software configuration skills and general game design skills. The software firm is portrayed as a *profit oriented business*, governed by industry forces such as *time to market pressures*, and cultural milieu characteristics such as the *individualistic* and *entrepreneurial* behavior as expressed by software developers (Sawyer, 2000). As suggested in the model, creation and transformation of community knowledge takes place in the interplay between the software community and the software firm. This creation and further transformation of knowledge is understood as consisting of three iterative and interrelated processes: knowledge building, knowledge elicitation, and knowledge exploitation. For me, these distinctions are primarily analytical as to facilitate an understanding of the way in which community knowledge use can improve PSD, and what repeated activities this entails for the software firm attempting to benefit from this knowledge.

As can be seen in the model, I view each cycle of community knowledge use as consisting of three processes. First, there needs to be community knowledge to be transformed, I.E. there is the need for processes supporting customers in their creation and sharing of situated knowledge. In this thesis, this process is referred to as *knowledge building*, denoting the *process of supporting software customers' creation and sharing of situated product knowledge*. In this, virtual communities work as enablers of knowledge creation and exchange between customers, and also between customers and developers within the software firm. In accordance with Boland and Tenkasi (1995), dynamic interaction within and between such communities is understood as important for new knowledge to emerge. However, and as illustrated in the model, knowledge building is understood as taking place mainly within the community of software customers. While the software firm can indeed support the initiation of this process, I view the building process itself as unfolding through software customers' mutual engagement in specific software products.

Second, while knowledge building activities support the creation and sharing of knowledge within customer communities, there need to be organizational activities for making sense of this knowledge, I.E. arrangements for sensemaking capabilities (Weick, 1979) within the software firm. In this thesis, this process is referred to as *knowledge elicitation*, denoting the *process of making sense of customer-generated product suggestions*. To succeed in this process there needs to be the appreciation and understanding of shared repertoires as expressed within the community. These repertoires include routines, words, tools, gestures, symbols, actions, and concepts that the community has produced or adopted in the course of its existence (Wenger 1998). In making sense of these styles by which community members express their forms of membership and ideas, there is the possibility for knowledge elicitation in terms of customer suggestions for software improvements. As illustrated in the model, knowledge elicitation takes place at the intersection of the community and the software firm. Here, the firm needs to understand and appreciate the repertoires as expressed by the community. Also, the community's willingness to share its knowledge with firm representatives is critical.

Third, there is the need for organizational implementation of the knowledge that has been built and elicited in previous processes. In my model, I refer to this process as *knowledge exploitation*, denoting the *process of transforming customer suggestions into software improvements.* To succeed in this, organizational resources for implementing customer suggestions are required as well as product development iterations that make possible for this transformation. As illustrated in the model, knowledge exploitation takes place exclusively within the software firm in order to implement customer suggestions into software improvements. Analytically, the process of knowledge exploitation can be seen as the third and final process of community knowledge use. However, in practice the processes of knowledge building, knowledge elicitation and knowledge exploitation are ongoing and closely interrelated. Furthermore, I view all three of them as necessary for the successful creation, understanding and use of community knowledge and hence, as central for improving PSD.

4.3 THE COMMUNITY USE MODEL

In an attempt to portray the PSD environment in which software communities and software firms are constituent parts in knowledge creation and transformation processes, I present the *community use model* (FIGURE 5). This model is to be interpreted as *a way of understanding* community use in PSD. For me, this understanding has emerged throughout my research process, and while components can be seen in the different phases of the research process, the model I propose here is the result of an emerging understanding of the research phenomenon. On the basis of theoretical as well as empirical insights, the model reflects the constituent parts and processes I perceive central for understanding community-based customer involvement in PSD. Hence, the model should be seen as portraying the conditions under which community knowledge may be built, elicited and exploited in the PSD environment, as well as how the results of these processes may affect future PSD environments in terms of software use and software firms.

In the model, *distributed software use*, I.E. situated knowledge as enacted in use by distributed software customers and *software firm environment*, I.E. characteristics imposed on the PSD environment in terms of industry, software development, cultural milieu and teams, are seen as conditions affecting the virtual community (ARROW A and ARROW B) and the knowledge creation and transformation processes as conducted within this. Together, these conditions, and the actors within these, are what constitute the virtual community in terms of software customers and software firm representatives using the community infrastructure for creation and transformation of *community knowledge*, I.E. knowledge building, knowledge elicitation and knowledge exploitation (ARROW C, ARROW D and ARROW E). If extracted from the community (ARROW F), community knowledge serves as input for improving PSD products and processes. Also, extracted community knowledge works as input in future software use (ARROW G), I.E. new use situations due to new software products, and in future software firm environments (ARROW H), I.E. new environmental conditions due to new software processes.

In understanding community use as portrayed in the community use model, there is the possibility to see how community knowledge is created within — and extracted from — customer communities. In this way, the model reflects my understanding of *community-based customer involvement in* PSD, intended to improve the difficulties with this as identified in research within the field (Sawyer, 2000; Grudin, 1991).

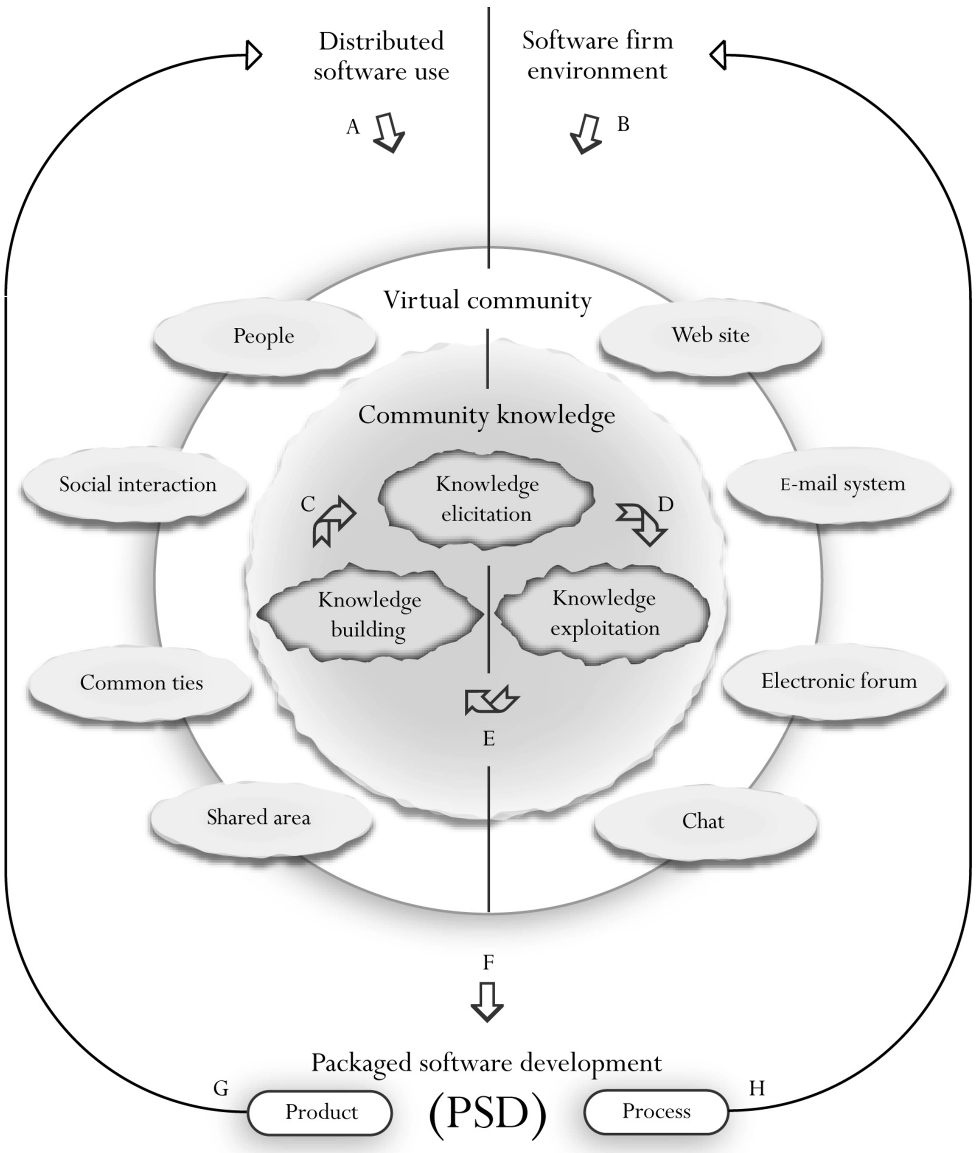


FIGURE 5. The community use model

5 Research contributions

Many are those who have highlighted the problems associated with generalizing the results of interpretive research in general and interpretive case studies in particular (Patton, 2002; Walsham, 1995). It is argued that the results only illuminate a particular situation or a very small number of cases (Patton, 2002). However, it is also recognized that certain kinds of small samples are selected and studied precisely because they have broader relevance

(Mintzberg, 1979; Patton, 2002). The view that also particulars and single cases contribute to general knowledge is expressed by Stake (1978) in saying that: *"Generalization may not be all that despicable, but particularization does deserve praise. To know particulars fleetingly, of course, is to know next to nothing. What becomes useful understanding is a full and thorough knowledge of the particular, recognizing it also in new and foreign contexts"* (Stake, 1978). Concurring with these authors, I believe that we should not be misled into too narrow a view of generalizations which readers can gain from interpretive case studies. In accordance with Walsham (1995), however, I do believe that my results here should be seen as tendencies rather than data for predicting future events. Indeed, while there will always be the attempt to benefit from customer involvement in terms of community use as reported on here, there will also be the situated challenges of software use, firm environments and community culture that will affect how this involvement can be achieved and hence, how community use can improve a specific PSD process. Therefore, my results should be seen as descriptive rather than normative. For instance, the conceptual model that I present should be interpreted as a way of understanding community use rather than predicting the way in which this should be undertaken.

In this section, I intend to summarize the contributions of this thesis. To do this, I have chosen to summarize the *thesis papers* (section 5.1) as each of these represents components of my emerging understanding of the research phenomenon. As a result of continuous iteration between empirical data and theoretical concepts, each paper elaborates on different aspects of the case, and together, they provide a comprehensive view of what I call community-based customer involvement. Also, *related papers* are listed (section 5.2) as a result of my research process. Following the presentation of the thesis papers and the listing of related papers, I discuss my theoretical contribution in terms of the *community use model* (section 5.3) and what this conception entails for *community-based customer involvement*.

5.1 THESIS PAPER OVERVIEW

In exploring subsets of the research phenomenon outlined in this thesis, each individual thesis paper contributes to the recognition of community use for improving customer involvement in PSD. As outlined below, different aspects of this theme have been elaborated upon in the papers and it is my intention that, taken together, these papers will provide the reader with a rich understanding of the particulars of this case. In summing up the empirical and theoretical insights from each paper there is the recognition of common themes that all contribute to what I term *community-based customer involvement*.

5.1.1 Internetworking with customers — PAPER 1

Henfridsson, O., and Holmström, H. (2002). Developing E-commerce in Internetworked Organizations — customer involvement throughout the value chain in the case of the online computer game Clusterball. DATA BASE — Special Issue on Developing E-Commerce Systems, Current Practices and State-of-the-Art. VOL. 33, NR. 4, PP. 38-50.

Paper 1 In building on empirical data from the first three phases of the study, I.E. the exploratory study, the in-depth study and the complementary data collection, this paper illustrates how network technologies allow for on-going interaction with customers and hence, for facilitating the involvement of these throughout the process of software development. Using Orlikowski's (1999) notion of *internetworked organizations*, we show how internetworking with customers benefits the process of computer game development in terms of better responsiveness to changing customer behavior and, ultimately, in achieving better computer game design. In the paper, both historical and social background to the case is given and we learn about Daydream as an innovative software firm with high reaching visions on outsourcing parts of development, evaluation, distribution and marketing to those consuming the products. In achieving this, technologies such as a customer relationship management (CRM) database and a virtual community are deployed, and in releasing Clusterball also as an online product, there is the recognition of how network technologies, and the internetworking as described by Orlikowski (1999), extend the notion of E-commerce by involving customers also in producing value.

However, while presenting customer involvement as beneficial for knowledge intense organizations, there is the recognition of challenges associated with this. In the paper, two challenges that need to be addressed when involving customers as co-producers in the software development process are outlined. These challenges are *customer role challenges*, and *sensemaking challenges*. In terms of customer role, the paper notes how involving customers in all stages of the value-adding process entails an increasing customer dependency that can be difficult to handle in an internetworked environment characterized by low switching costs in the firm-customer relationship. In addition, sensemaking challenges, I.E. capabilities to transform customer input into meaningful software improvements, are identified as difficult to manage in the context of E-commerce that traditionally has been dominated by standards for enabling exchange of information between seller and buyer. In contrast to this, this paper calls for a *contextual approach* for involving customers and it is concluded that both challenges need to be further explored in future research.

5.1.2 *Customer knowledge in software development* — PAPER 2

Holmström, H. (2001). Virtual Communities as Platforms for Product Development — an interpretive case study of Customer Involvement in Online Game Development. *In Proceedings of* ICIS *2001, (22nd International Conference on Information Systems),* December 16-19, New Orleans, LA, USA.

Paper 2 Adding to the empirical insights from paper one, and particularly to the call for a contextual approach for involving customers in software development, this paper discusses the use of virtual communities for testing, distributing, redesigning and evaluating a computer game. Based on an understanding of virtual communities as infrastructures for improving customer relationships, this paper adheres to the case study research on software communities in which interaction becomes a prerequisite for satisfying also commercial needs. As in paper one, empirical data originates from the first three phases of the Daydream study and while customer involvement is still the focus, this paper specifically recognizes *customer knowledge* and how this knowledge may be utilized in each phase of the software development process. On the basis of empirical data, it is noted that many suggestions for software improvements come from customers and that customer knowledge is especially important in the process of testing and evaluating Clusterball. Realizing this, there are reasons to believe that customer knowledge proves particularly useful for *improving and maintaining* software. In exploring this further, this paper calls for research on (1) structure redefinition, I.E. the transformation of the software development process due to continuous customer involvement, and (2) role definition, I.E. the understanding of customer involvement as transforming the role of customers.

5.1.3 Customer role ambiguity — PAPER 3

Holmström, H., and Henfridsson, O. (2002). Customer Role Ambiguity in Community Management. *In Proceedings of* HICSS *35 (35th Hawaii International Conference on System Sciences),* January 7-10, Big Island, Hawaii.

Paper 3

As recognized in the first two papers of this thesis, there are challenges associated with the enhanced customer role that is discernible when involving customers throughout the value-adding process of software development. Hence, the first two papers emphasize the need for further research on this topic. To cater for this, the third paper further explores *customer role ambiguity* as a critical issue for successful community management. In defining community management as the activity of establishing, maintaining and re-producing virtual communities, this paper examines the delicate opportunity of having customers act in the role of producers by devoting time and energy to value-adding activities such as product development and marketing without monetary compensation; on the other hand, the customers act in the role of consumers of the value produced by these activities. However, while this opportunity is outlined as promising and relevant especially for firms aiming at the business-to-consumer market, it is associated with customer role ambiguities in terms of individual customers' uncertainty about the expectations surrounding his or her role, or in terms of interference with goal accomplishment. On the basis of empirical examples, three different customer role ambiguities are identified: (1) role absorption, I.E. difficulties for Clusterball ambassadors to handle the role as both player and company representative, (2) business model violation, I.E. difficulties with having customers contribute to value-adding development activities without violating the commercial business model and the way in which profit was to be made, and (3) non-organizational network elements, I.E. difficulties with telling where information originated since parts of game diffusion was handled outside traditional organizational boundaries. In discussing these ambiguities and the difficulties they entail in relation classic dimensions of business organization, I.E. trust building, business modeling, and organizational transformation, we further explore how customer role ambiguity adds new dimensions to these traditional elements of business organization. In concluding the paper, we suggest that an understanding of customer role ambiguities and the consequences these entails for classic dimensions of business organization is critical for making product-centered communities a viable alternative to traditional software development.

5.1.4 *Distributed software development approaches* — PAPER 4

Holmström, H. (2003). The Distributed Nature of Software Development — a comparison of three development approaches. *In Proceedings of* PACIS *2003 (Pacific Asia Conference on Information Systems),* July 11-13, Adelaide, Australia.

Paper 4 This paper broadens the scope of the thesis with the purpose of understanding the software development environment in which PSD is conducted. Here, my focus is changes that are discernible in today's software development environment. As suggested in the paper, software development is to a greater extent becoming a distributed process and there is the need for development approaches that take into consideration the distributed environment in which software developers and software customers communicate and coordinate their work. On the basis of empirical results from the first three phases of the Daydream study, as well as complementing data from a secondary analysis of two published case studies, three approaches to distributed software development are explored, I.E. *global software development* (GSD), *open source software development* (OSS) and *community-based software development* (CSD). In a comparison, it is suggested that these approaches embrace differences in terms of (1) *nature of development approach*, (2) *communication structure*, and (3) *coordination mechanisms*, where the CSD approach is especially relevant since this is the approach discernible in the empirical case reported on in this thesis. In being recognized as an approach adopted by companies aiming to infuse OSS features into traditional software development, CSD is described as a hybrid approach characterized by formal and informal structures and suitable for traditional software firms willing to involve its customers in their PSD processes. As concluded in this paper, the identification of these approaches and the differences they embrace is helpful for both researchers and practitioners for recognizing the different approaches to distributed software development that exist, and for creating an understanding of the types of development situations in which these can be successfully applied.

5.1.5 *Customer involvement in packaged software maintenance* — PAPER 5

Holmström, H., and Fitzgerald, B. (forthcoming). Virtual Community Use for Packaged Software Maintenance. *Accepted for publication in the Journal of Organizational Computing and Electronic Commerce* — Special Issue on « Virtual Communities and Personalization in E-commerce ».

Paper 5 As recognized in the second paper, customer involvement — in terms of customer knowledge — is of great importance especially in the process of improving and maintaining packaged software. To further elaborate on this, this paper explores the specific use of customer knowledge for *packaged software maintenance*. On the basis of empirical data from all four phases of the study, I.E. the exploratory study, the in-depth study, the complementary data collection and the follow-up study, the paper investigates to what extent and in what situations customer knowledge is reflected in different categories of software maintenance. Three categories of software maintenance are explored, I.E. *corrective, adaptive and perfective maintenance*, and it is concluded that customer knowledge, as generated within the virtual community, is reflected mainly in the first two categories of maintenance. Here, customers contribute in the process of software fault repair and software adaptation. As can be seen in the different patch specifications that are outlined in the paper, customer suggestions are evident in each new version of the game. However, while customer knowledge is reflected also in the in the third category of software maintenance, I.E. perfective maintenance, this is also the category where strong in-house ideas as well as external requirements put restrictions to the extent to which customer suggestions are considered. As a result, customer knowledge is exploited for the purpose of corrective and adaptive packaged software maintenance, but only to a limited extent exploited for the purpose of perfective packaged software maintenance.

5.1.6 *Community knowledge for improving* PSD — PAPER 6

Holmström, H., and Henfridsson, O. (submitted). Improving Packaged Software Through Online Community Knowledge. *Submitted to an international* IS *journal.*

Paper 6 The sixth and final paper uses empirical data from all four phases of the study in order to explore PSD as a knowledge intensive activity unfolding at the intersection between commercial software firm practices and voluntary community participation. In bringing findings from previous papers together, we examine benefits and limits of community use in PSD. In the paper, the challenges as outlined in previous papers are further elaborated upon and on the basis of theoretical concepts as well as empirical insights, we present a conceptual model for understanding community use in PSD. This model — *the community use model* — describes such use as three iterative and interrelated processes unfolding at the intersection between the software firm and the software community. These processes are (1) knowledge building, I.E. the process of supporting software customers' creation and sharing of situated product knowledge, (2) knowledge elicitation, I.E. the process of making sense of customer-generated input and transforming this into software improvements, and (3) knowledge exploitation, I.E. the process of implementing elicited customer knowledge into software improvements. Furthermore, the paper highlights the inherent tension between the motivational structures of commercial software firms and those of voluntary community participation — a tension that is understood as a true challenge of PSD.

5.2 RELATED PAPERS

In addition to the six thesis papers, there is a set of related papers. While these papers have not been included in the thesis, they have all been important to me and my work for many reasons. First, among these there are several IRIS (Information Research in Scandinavia) publications. The opportunity for me to present these papers in a constructive atmosphere at the IRIS conferences has been of great help to me as a PhD student. Not only has it given me experience in presenting my work, but also it has introduced me to the Scandinavian community of researchers within the field of IS. As a yearly meeting-place for doctoral students as well as senior researchers, the IRIS conferences has introduced me to a large network of IS researchers of which many have proven to be of great importance to my work.

Second, many of these papers represent ideas that have been further elaborated upon in the thesis papers. In this, they have constituted an important initial step in my research and can be seen as the seeds to much of what have finally been selected as to represent my work as a PhD student. Finally, the related papers reflect my research process. While much of what has been included in the thesis reflect what was written during the recent years (2002-2004), these related papers adhere to the understanding of this process as a longitudinal project during which much work was done also during the time for the empirical study (2000-2002). The related papers are listed below:

(1): Holmström, H. (2000). Getting to know your customers: Implications of virtual communities in on-line commerce. *In Proceedings of* IRIS *23 (Information Research in Scandinavia),* August 12-15, Uddevalla, Sweden.

(2): Holmström, H. (2001). Virtual Communities as Mediators of Customer Expertise — an interpretive case study of interactive product development in on-line entertainment. *In Proceedings of* IRIS *24*, *(Information Research in Scandinavia)*, August 11-14, Ulvik, Hardanger, Norway.

(3): Henfridsson, O., Holmström, H., and Hanseth, O. (2001). Better safe than Sorry? In search of an Internet business model in on-line entertainment, *In Proceedings of IFIP 8.2 WG Conference*, July 27-29, Boise, USA.

(4): Holmström, H. (2002). Virtual Communities in Distributed Software Development. *In Proceedings of* IRIS *25, (Information Research in Scandinavia),* August 10-13, Bautahøj, Denmark.

(5): Holmström, H., and Henfridsson O. (2003). Towards a Community-based Approach to User Participation. *In Proceedings of* IRIS *26, (Information Research in Scandinavia),* August 9-12, Porvoo, Finland.

(6): Holmström, H. (2004) Virtual Communities for Software Maintenance. *In Proceedings of* HICSS *37 (37th Hawaii International Conference on System Sciences),* January 5-8, Big Island, Hawaii.

(7): Holmström, H. (2004) Involving Distant Users in Packaged Software Development: A User Community Approach. *In Proceedings of* IRIS *27 (Information Research in Scandinavia), and presented as a « plenary paper » at the conference.* August 14-17, Falkenberg, Sweden.

5.3 THE ROLE OF COMMUNITY USE IN PSD

As recognized in this thesis, PSD is inherently different from custom IS development. Here, software is targeted to a mass market of geographically distributed customers and while custom IS benefit from well-established development methods, PSD processes are regarded immature and less likely to adhere to sequential development with separated development phases (Sawyer, 2000). Above all, customer involvement is yet to gain momentum, and for the time being, many PSD processes suffer from lack of customer involvement due to distant (Sawyer, 2000), and unknown (Grudin, 1991) customers. As recognized by Keil and Carmel (1995), there is a bewildering array of customer-developer links from which to chose, E.G., support lines, surveys, user groups, marketing and sales and trade shows. However, since many of these are indirect links in which customers and developers communicate through intermediaries or customer surrogates, they are difficult to rely too hard upon. While there is not the question of *whether* customers should participate in the PSD process, there is the question of *how* they should efficiently do this (Sawyer, 2000; Keil and Carmel, 1995). In an attempt to solving this problem, this thesis has explored the role of *virtual communities* for involving distributed customers in PSD. As the motivation for this, there is my belief that better utilization of knowledge inherent in customer communities could benefit product development processes. As recognized by Orlikowski (2002), knowledge is grounded in what people do and what they experience in their every-day situations. In these situations, knowledge creation is an ongoing social accomplishment that is constituted and reconstituted only in everyday social practices and in everyday social dealings with like-minded people (Brown and Duguid, 2001). Furthermore, knowledge is something that is achieved not given, and continuous negotiation is therefore critical in achieving this (Wenger, 1998), as well as systems and processes for successful integration of this into the product development process (Nambisan, 2002).

In accordance with this notion of knowledge and the way in which it is manifested in distributed customer communities, an important contribution of this thesis is a model for understanding *community use* (FIGURE 5 in section 4.3). This model is a conceptual understanding representing environmental conditions as well as internal knowledge creation and transformation processes important for community-based customer involvement in PSD. The *community use model* reflects the constituent parts and processes that I perceive central for understanding how community use can improve customer involvement in PSD. Hence, the model should be seen as portraying the conditions under which community knowledge may be built, elicited and exploited in the PSD environment, as well as how the results of these processes may affect future PSD environments in terms of software use and software firms. In this model, *distributed software use*, I.E. situated product knowledge as enacted in use by software customers (for example graphic skills, hardware and software configuration skills and gaming skills), as well as *software firm environment* characteristics, I.E. conditions imposed on PSD due to its industry, software development, cultural milieu and team characteristics work as input in the cyclical processes of *knowledge building*, *knowledge elicitation* and *knowledge exploitation*. Here, knowledge building refers to the process of supporting software customers' creation and sharing of situated product knowledge. As understood in the community use model, this process takes place within the community of software customers and hence, is relatively independent of what a software firm does to promote it. While the software firm can indeed support and facilitate the initiation of this process (see E.G., the creation of the Clusterball website and the appointment of a community manager and Clusterball ambassadors in the Daydream case), I view the building process itself as unfolding through software customers' mutual engagement in a specific software product. In other words, given the existence of interesting software, the process of knowledge building unfolds relatively independently as a result of community commitment and engagement. However, since knowledge building is seen as such an important impetus to the later processes of knowledge elicitation and knowledge exploitation, it is vital that the software firm engages as facilitator in this process, realizing the opportunities that exist.

Following the process of knowledge building, knowledge elicitation refers to the process of making sense of customer-generated product suggestions. Here, an appreciation and understanding of shared repertoires such as routines, words, tools, gestures, symbols, actions, and concepts as expressed within the community is needed. In making sense of the styles by which community members express their membership and ideas as community participants, there is the possibility for knowledge elicitation in terms of customer suggestions for software improvements. As understood in the community use model, knowledge elicitation takes place at the intersection of the software community and the software firm. Here, the software firm needs to develop a sensemaking capability for understanding and appreciating the repertoire as expressed by the community. Also, the process presupposes the community's willingness to share its knowledge with firm representatives interested in eliciting this. While the knowledge elicitation process is suggested for making sense of customer suggestions, this can be done in many different ways. As experienced in the Daydream case, a simple form of knowledge elicitation may consist of monitoring the community forum for postings on software operability problems. Doing this, the community is used more or less as a bug-reporting system and the knowledge elicitation process is not so much of a sensemaking process but instead a straight forward process of collecting as many customer suggestions as possible for solving minor software problems. At a more advanced level, however, arrangements for internalizing (and in later processes implementing) more substantial customer feedback are necessary, indicating the need for a boundary spanning role, see E.G., the community manager role in the Daydream case, grounded in community participation as well as in firm practices.

In concluding the community knowledge use cycle, knowledge exploitation refers to the process of transforming customer suggestions into software improvements. To succeed in this the software firm needs to develop organizational resources for dealing with customer suggestions and transforming these into software improvements. As understood in the community use model, knowledge exploitation takes place exclusively within the software firm in order for software developers to implement customer suggestions and hence, benefit from community knowledge in their software development processes. While the term « exploitation » may denote an instrumental process of commercial utilization of co-produced community knowledge, this is — from a software firm perspective — what often takes place. While community building processes intended to nurture, support and stimulate community culture are indeed important for software firms aiming for community-based customer involvement, commercial interests will almost always be prioritized in the later stages of community knowledge use. Therefore, knowledge building and elicitation processes may support the creation and transformation of customer suggestions, while these — in cases where they contradict commercial interests — will most probably be neglected by the software firm. While such prioritizing is evident in most development projects, it might be more difficult to explain to devoted community members looking for their suggestions to be implemented in the software as encouraged in the knowledge building and elicitation processes. Clearly, knowledge exploitation is the most contentious process within the community knowledge use cycle and as recognized in the Daydream case, there is the delicate situation of having customers as both producers and consumers of value.

In understanding knowledge building, knowledge elicitation and knowledge exploitation as processes unfolding at the intersection between the software firm and the software community, my research recognizes the different motivational structures of commercial software firms and those of voluntary community participation (PAPER 6). In the software community, we find software customers interested in a particular software product using the community for sharing knowledge around this product. Here, social interaction around a common interest is of primary interest, and the virtual community infrastructure is used as an enabler for social interaction. In the software firm, on the other hand, there are profit oriented interests in an environment characterized by tight time to market pressures. Also, software developers are known as individualistic and entrepreneurial (Sawyer, 2000), using virtual community infrastructure for exploiting knowledge inherent in the customer community. While this difference in motivation might indicate a tension between the two, the understanding of knowledge as enacted in moments of action and emerging only through ongoing relationships of context (Orlikowski, 2002), makes them deeply dependant on each other, as well as central to the context of PSD. The result of these processes is reflected in PSD products in terms of new software releases, I.E. software improvements, and in PSD processes in terms of new ways of working, I.E. re-organization of development activities. Furthermore, the model conceptualizes the results of knowledge exploitation as cyclical in that its output, I.E. PSD product and process improvements, are understood as affecting future distributed software use (E.G., future product knowledge creation among distributed customers) and future software firm environments (E.G., future firm conditions as reflected in market share, development methods, organizational structures and formation of project groups). In this, the model portrays *community use as a continuously ongoing interplay* between the software firm and the software community in which knowledge creation and transformation processes are a result of commercial firm interests as well as voluntarily community participation.

So, in understanding community use as consisting of knowledge building, knowledge elicitation and knowledge exploitation, and as taking place in a development environment influenced by commercial ideals as well as more altruistic community ideals — what is the role of virtual communities for improving PSD? To answer this question, we need to return to the research phenomenon outlined in the beginning of this thesis. Here, I urge for exploring the challenges associated with involving customers in PSD. While there are indeed approaches for involving customers, these are seen as insufficient and difficult to apply in the PSD environment characterized by distant (Sawyer, 2000) and unknown (Grudin, 1991) customers. In meeting these challenges, I suggest virtual community use as a viable approach. In understanding virtual communities as existing at the intersection of technical and social systems (Stanoevska-Slabeva and Schmidt, 2001), I suggest that a community-based approach to customer involvement involves the:

(1) recognition of virtual communities for bringing distributed customers and customer knowledge together for the purpose of customer involvement in distributed development environments, I.E. an understanding of the *technical aspects* of communities, and the…

(2) recognition of virtual communities for establishing, maintaining and reproducing relationships important for the purpose of knowledge creation and transformation processes, I.E. an understanding of the *social aspects* of communities.

In understanding the community-based approach as consisting of both technical and social aspects, there is the possibility to see how this approach might be able to address the research phenomenon in that it embraces:

(1) technical aspects catering for the distributed development environment and the challenges this introduces for customer involvement, I.E. the difficulty identified with having *distant customers* (Sawyer, 2002), and…

(2) social aspects catering for the community-building process and the challenges this introduces for customer-developer relationships to be established, I.E. the difficulty identified with having *unknown customers* (Grudin, 1991).

5.4 OPPORTUNITIES AND CHALLENGES WITH COMMUNITY USE

In facilitating an understanding of how community use can improve PSD, and in recognizing the cyclical processes of knowledge building, knowledge elicitation and knowledge exploitation, it is my intention that the community use model contributes to an enhanced appreciation of community use in PSD. However, I have come to understand that the phenomenon this model depicts is far from simple but instead associated with complexities in terms of *opportunities* and *challenges* affecting the way in which it will be interpreted and hence, understood. Below, the opportunities and challenges that I encountered in my research are outlined, categorized in accordance with the knowledge creation and transformation processes as identified in the community use model, I.E. knowledge building, knowledge elicitation and knowledge exploitation.

• *Internetworking opportunities* (PAPER 1 and PAPER 2). One immediate opportunity in the process of knowledge building is the possibility to use the community infrastructure for internetworking with distributed customers. In facilitating connectivity between software customers and software developers, virtual communities can be seen as an efficient means in avoiding filtering or distortion of knowledge that may occur when using intermediaries or customer surrogates as customer representatives. As a result of this, community-based knowledge building makes possible for the software firm to establish a direct link to its distributed customer community. While community knowledge is something that resides naturally within a community, the process of knowledge building contributes to the view of knowledge co-creation as central for a new customer—producer relationship as is evident in many product development industries of today.

• *Enhanced customer role opportunities* (PAPER 3 and PAPER 4). The knowledge building process reflects an enhanced customer role in which customers are appreciated not only as consumers of value but also as producers of value. Recognizing the knowledge inherent in the customer community, software customers are seen as co-producers of knowledge important for software development. As a result of knowledge building processes, customers contribute to a variety of development activities including, for example, opportunities to make design choices, prioritization of product features, concept testing and the possibility to influence product customization. Thus, community-based knowledge building makes possible for expanding the role of customers to become also producers of knowledge in PSD processes.

• *Customer role challenges* (PAPER 1 and PAPER 3). In association to the enhanced customer role, customer role challenges are discernible. In taking the role as co-producers of knowledge, my research shows customers as increasing their control and influence on the PSD process and hence, as challenging the traditional producer-consumer relationship. As a result of this, community-based knowledge building brings with it customer dependency leading to project uncertainty. Due to low switching costs and the fear of having customers abdicate their role as knowledge co-creators, thereby disrupting the development process, knowledge building needs to be carefully undertaken. Also, customer role ambiguity might arise, referring to the uncertainty customers and vendors might feel about the expectations surrounding the enhanced customer role as experienced in the process of community-based knowledge building.

• *Contextual approach opportunities* (PAPER 2). Community-based knowledge elicitation allows for a contextual approach in which the interpretation of knowledge is not dominated by standards but instead on situated capabilities reflecting in-house development visions as well as current business needs. For this purpose, my research shows customer interaction as pivotal to the knowledge elicitation process. In providing functionality such as electronic forums and chats, community technology allows for a setting in which knowledge can be informally elicited in a way that has been found more beneficial than formal knowledge elicitation provided by, for example, structured inquiry tools.

• *Sensemaking challenges* (PAPER 1 and PAPER 6). In the process of knowledge elicitation, sensemaking challenges are evident. While customers might indeed be willing to share their knowledge there is the need for a sensemaking capability within the software firm, attentive to the knowledge built in light of perceived business needs. In my research, this sensemaking capability was expressed in terms of new organizational roles, E.G., the community manager role. However, while such arrangements are indeed useful, the risk of capturing only a fraction of the knowledge built due to, for example, individual preferences and prerequisites is evident. Also, this process entails the need for selecting representative customer suggestions, I.E., the software firm's ability to make sense of a wide range of customer input for exploitation of only competitive suggestions.

• *Maintenance opportunities* (PAPER 5). For the purpose of exploitation, my research shows community knowledge as particularly useful for software maintenance. Using the tripartite typology of corrective, adaptive, and perfective maintenance, exploitation of community knowledge proves especially valuable for the categories of corrective and adaptive maintenance. As illustrated in the Daydream case, the community allowed for customers to be efficiently involved in troubleshooting and in software fault repair. As a result of this, customers were involved in the knowledge exploitation process as accomplished by the software firm also after the product was introduced to the market.

• *Implementation challenges* (PAPER 5 and PAPER 6). In relation to exploitation of community knowledge, implementation challenges are explicit. While being exploited in both corrective and adaptive software maintenance, customer suggestions are more difficult to implement in the category of perfective maintenance. Despite customer input, E.G., customer requests and suggestions, these are often ignored contrary to external requirements or in-house ideas. As a result of this, knowledge exploitation brings with it the need for clarifying development objectives, I.E., the software firm's ability to communicate to what extent community knowledge is considered in each category of software maintenance.

6 Conclusions

In viewing community-based customer involvement as an approach for solving the difficulties with having distant and unknown customers, my research portrays interesting features of this approach that will influence our general perception of PSD practice. I suggest that the approach I outline will alter the way in which we understand and reflect upon PSD products and processes. First, my research depicts community-based customer involvement as a hybrid between traditional software development and open source software (OSS) development. The OSS model is a new way to develop software and in this, a model that is setting the stage for a structural change in the software development industry. However, while there are significant benefits of OSS development, there is not always the possibility for traditional software firms to deploy this model of distributed development. Most often, for-profit organizations, such as for example software vendors, have difficulties in building business models around the OSS paradigm (Sharma ET AL, 2002). Instead, there is the challenge of finding ways to incorporate aspects of the community culture into traditional software development processes and in this way infuse characteristics of the OSS paradigm into traditional software development. In my view, this is the incentive of the community-based approach that I present. As can be seen in my research, community-based customer involvement allows for reduced development time (due to customers helping the developers with bug tracing and bug correcting activities), improved quality (due to community criticism and feedback), reduced cost (due to outsourcing parts of development on customers), increased developer loyalty (due to the view of customers as not only consumers but also producers of value), and increased developer talent pool (due to continuous enrolment of new community members). These characteristics can all be found within the OSS paradigm, and what is interesting here is that — in adopting a community-based approach to customer involvement — these characteristics are transferable to traditional software firm settings as well. As illustrated in the Daydream case, the community-based approach allows for a traditional software firm, in this case Daydream, to infuse characteristics found in OSS development into its PSD process and in this way, foster an environment similar to OSS. In other words, the community-based approach suggests that traditional software firms can reap many of the advantages as experienced within OSS development without ever deploying this model. For example, organizations like Hewlett Packard, IBM, Intel, Sun Microsystems, ETC., have already taken steps to use communities as a way to incorporate elements of OSS into their software development processes. This, I think, indicates the belief in communities as beneficial for involving customers in the development process of packaged software products and this is also what can be seen in the research I report on here.

Furthermore, my research highlights the commercial interest in community use. While there are voices emphasizing the social fabric of community (Schwen and Hara, 2003), recognizing the informal, nonhierarchical and social frame for considering these, the utilization and exploitation of community knowledge, as reported on here, suggests a kind of detached relationship to the complex practices as evolving within a community in the course of its time. As in OSS development where it might be hard to maintain the delicate balance between self-deprecation and modesty and the egoistic motivations that inevitably arise in a reputation-based culture, community-based customer involvement implies a struggle with balancing between having customers act in the role of producers by devoting time and energy to value-adding activities without any monetary compensation, and at the same time, act in the role of consumers on which profit is to be made. However, and in similar with OSS culture, the community culture as experienced in the Daydream case is undeniable a reputation-based one, attracting individuals aiming for status and recognition from those within the community. In this, it seems to me that commercial exploitation of community knowledge is not that controversial. Rather, community members rush from seeing their suggestions being implemented in the software and one of the most important incentives for community membership is the ego gratification as experienced when having contributions acknowledged by software firm representatives. As in OSS, this phenomenon can be described as an attention economy (Bergquist and Ljungberg, 2001), revealing exploitation of community knowledge as appreciated rather than controversial.

Finally, my research portrays a new way of looking at software development in general and software maintenance in particular. Traditionally, the conception of software development and maintenance is that of tedious and time-consuming processes. Thus, the opportunity to have parts of development outsourced to customers is indeed attempting. In this, the community-based approach I suggest bodes well as a model. Furthermore, despite research which suggests that software maintenance consumes between 40 to 75 percent of the total resources in software development (see E.G. Alkhatib, 1992; Lientz and Swanson, 1980), maintenance appears not to be highly regarded by software developers. As can be seen in my research, however, software maintenance is undertaken as a mutual commitment between software customers and software developers. While customer-developer relations in traditional software maintenance often become frayed, the positive atmosphere as reported on here makes community-based customer involvement a prevailing idea. However, the word « maintenance » can be misleading when referring to adaptive changes such as those reported on here. While « maintenance » gives the impression of software as degraded and in need to be refurbished to its original condition, this is not true for the type of software I have studied. Software products, such as a computer game, do not degrade. Instead, they remain the same, while the environment and equipment around the product continuously changes. Thus, adaptive maintenance, as experienced in the Daydream case, is really a process of upgrading or improving the software to meet the needs of customers and surrounding product environments. Hence, in the case of community-based customer involvement I report on, software improvement is a better word for describing what has traditionally been referred to as software maintenance, an idea I suspect is valid also when referring to the life cycles of other packaged software products.

To conclude, community-based customer involvement can be seen as a comprehensive approach for understanding community knowledge as critical for improving PSD. It is my suggestion that, in involving customers and their situated knowledge in PSD, both products and processes will improve. However, this is a complex process consisting of knowledge creation and transformation processes taking place at the intersection of software communities and software firms. As can be seen in my research, these parties are closely interrelated and while driven by different motivational structures, they are indeed interdependent for the successful completion of community knowledge use. In recognizing external conditions as well as internal processes important for community use, it is my belief that the community-based approach I present will not only add to the question of *whether* customers should be involved in PSD but, in conceptualizing the role of virtual communities, add to the question of *how* they could be involved.

References

Abrahamsson, P., Warsta, J., Siponen, M., and Ronkainen, J. (2003). New Directions on Agile Methods: a comparative analysis. In Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, PP. 244-254.

Abts, C. (2002) COTS-Based Systems (CBS) Functional Density — A Heuristic for Better CBS Design. *In Proceedings of* COTS*-Based Software Systems*, First International Conference, ICCBBS 2002, Orlando, FL, USA, February 4-6.

Alkhatib, G. (1992). The maintenance problem of application software, Journal of Software Maintenance: Research and Practice, VOL. 1, pp-83-104.

Archer, S. (1988). Qualitative research and the epistemological problems of the management disciplines. In Competitiveness and the Management Process, Pettigrew, A. (ED.), PP. 265-302. Basil Blackwell, Oxford.

Avison, D.E., and Fitzgerald, G. (2003) *Information Systems Development: Methodologies, Techniques and Tools*. McGraw Hill: London.

Avison, D.E. and Taylor, V. (1997). Information systems development methodologies: a classification according to problem situation. *Journal of Information Technology*. VOL. 12, PP. 73-81.

Avison, D.E., and Wood-Harper, A.T. (1986). Multiview — An Exploration in Information Systems Development. *Australian Computer Journal* 18(4), PP. 174-179.

Baskerville, R., and Pries Heje, J. (2001). A Multiple-Theory Analysis of a Diffusion of Information Technology Case. *Information Systems Journal*, Volume 11, PP. 1-32. Blackwell.

Barki, H. and Hartwick, J. (1994). Measuring user participation, user involvement and user attitude. MIS *Quarterly*, PP. 59-82.

Barnes, B. (1983). On the conventional character of knowledge and cognition. In K.D. Knorr-Cetina and M. Mulkay (EDS.), *Science Observed*, London, UK: Sage Publications.

Baym, N. (1998). The Emergence of On-line Community. *In Jones, S. (*ED.*) CyberSociety 2.0: Revisiting computer-mediated communication and community* (PP. 35-68), Newbury Park, CA: Sage.

Beyer, H., and Holtzblatt, K. (1998). *Contextual Design; Defining Customer-Centered Systems*. Academic Press: London.

Boland, R.J., and Tenkasi, R.V. (1995). Perspective Making and Perspective Taking in Communities of Knowing. *Organization Science*, VOL. 6(4), PP. 350-372.

Bostrom, R.P., and Heinen, J.S. (1977). MIS Problems and Failures: A Socio-Technical Perspective. MIS *Quarterly*, September.

Butler, B.S. (2001). Membership size, communication activity and sustainability: a resource-based model of online social structures, *Information Systems Research*, VOL. 12, NO. 4, PP. 346-362.

Carmel, E. (1997). American Hegemony in Packaged Software Trade and the « Culture of Software ». *The Information Society,* VOL. 13, PP. 125-142.

Carmel, E., and Becker, S. (1995). A process model for packaged software development. IEEE *Transactions on Engineering Management*, VOL. 41(5), PP. 50-61.

Checkland, P. (1981). *Systems Thinking, Systems Practice*. Wiley, Chichester.

Clegg, C.W., Waterson, P.E., and Axtell, C.M. (1996). Software development: knowledge-intensive work organizations. *Behaviour & Information Technology*, VOL. 15(4), PP. 237-249.

Clegg, C.W., Waterson, P.E., and Axtell, C.M. (1997). Software development: some critical views. *Behaviour &Information Technology*, VOL. 16(6), PP. 359-362.

Curtis, P. (1992). Mudding: Social Phenomena in Text-Based Virtual Realities. *Intertek 3.3* (winter), PP.26-34.

Dibbel, J. (1998). *My tiny life — crime and passion in a virtual world*. New York: Henry Holt & CO.

Divitini ET AL. (1998). Internet-based Groupware for User Participation in Product Development. CSCW' 98/PDC' 98 *Workshop Proceedings*, Seattle, USA.

Dubé, L. (1998). Teams in packaged software development. *Information Technology & People*, VOL. 11(1), PP. 36-61.

Ehn, P. (1993). Scandinavian Design: On Participating and skill. In D. Schuler and A. Namioka (EDS.). *Participatory Design: Principles and practices*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, PP. 41-77.

Feller, J. & Fitzgerald, B. (2002). *Understanding Open Source Software Development*. Addison-Wesley, London.

Fernback, J. (1997). The individual within the collective: Virtual ideology and the realization of collective principles. In S. Jones (ED.), *Virtual Culture: Identity and Communication in Cybersociety*, (pp. 36-54). London: Sage.

Finch, B.J. (1999). Internet discussions as a source for consumer product customer involvement and quality information: an exploratory study. *Journal of Operations Management*, VOL. 17, PP. 535-556.

Fitzgerald, B. (1996). Formalized systems development methodologies: a critical perspective. *Information Systems Journal*, VOL. 6(1), PP. 3-23.

Fitzgerald, B. (1997). A preliminary investigation of RAD in Practice,. In Wood-Harper, A. T., Jayaratna, N., and Wood, J. (EDS.), *Methodologies for developing and managing Emerging Technology Bases Information Systems,* Springer-Verlag, UK, PP. 777-87.

Flaatten, P., McCubbrey, D., O'Riordan, P., and Burgess, K. (1989). Foundations of Business Systems, Chicago: Dryden Press.

Franz. C., R. and Robey. D. (1986). Organizational context, user involvement and the usefulness of information systems. *Decision Sciences* (17), July, PP.329-356.

Fukushima, T. and Martin, D. (1998). SMART Ideas as a tool for user participation in product development. In Divitini ET AL. 1998, CSCW' 98/PDC' 98 *Workshop Proceedings*, Seattle, USA.

Gallivan, M. (2001). Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal*, 11, PP. 277-304.

Gillham, B. (2000). *Case study research methods*. Continuum: New York.

Greenbaum, J., and Halskov Madsen, K. (1993). Small Changes: Starting a participatory design process by giving participants a voice. In D. Schuler and A. Namioka (EDS.). *Participatory Design: Principles and practices. Hillsdale*, New Jersey: Lawrence Erlbaum Associates, PP. 289-299.

Greenbaum, J., and Kyng, M. (1991). *Design at Work. Cooperative design of computer systems.* Hillsdale N.J. Lawrence Erlbaum.

Grudin, J. (1991). Interactive systems: Bridging the gaps between developers and users. IEEE *Computer* 24(4), PP. 59-69.

Grudin, J. (1993). Obstacles to participatory design in large product development organizations. . In D. Schuler and A. Namioka (EDS.). *Participatory Design: Principles and practices.* Hillsdale, New Jersey: Lawrence Erlbaum Associates, PP. 99-123.

Gronbaek, K., Grudin, J., Bodker, S., and Bannon, L. (1993). Achieving Cooperative System Design: Shifting From a Product to a Process Focus. In Schuler, D., and Namioka, A. (EDS.), *Participatory Design: Principles and practices.* Lawrence Erlbaum Associates: New Jersey.

Hagel, J., and Armstrong, A. (1997). *Net Gain — expanding markets through virtual communities.* Boston, MA: Harvard Business School Press.

Hamman, R.B. (2001). Computer Networks Linking Network Communities. In Werry, C., and Mowbray, M. (EDS.). *Online Communities.* Prentice Hall: London.

Henfridsson, O. (1999). IT-*adaptation as sensemaking — inventing new meaning for technology in organizations.* Doctoral thesis, Department of Informatics, Umeå University, Sweden.

Henson, K. and Hughes, C. (1991) A two-dimensional approach to systems development. *Journal of Information Systems Management*, PP. 35-43.

Hillery, G.A. (1955). Definitions of Community: Areas of Agreement. *Rural Sociology*, VOL. 20, PP. 111-123.

Holtzblatt, K., and Jones, S. (1993). Contextual Inquiry: A Participatory Technique for System Design. In Schuler, D., and Namioka, A. (EDS.), *Participatory Design: Principles and practices.* Lawrence Erlbaum Associates: New Jersey.

Hummel, J., and Lechner, U. (2001). Communities — The role of technology. In *Proceedings of the 9th European Conference on Information Systems.* 2001.

Ives, B. and Olson. M. H. (1984). User involvement and MIS success: A review of research. *Management Science* (30:5), May, PP. 586-603.

Jones, J.C. (1988). Softecnica. In Thackara, J. (ED.) *Design after modernism: Beyond the object*, PP. 216-226. London: Thames & Hudson.

Jorgensen, N. (2001). Putting it all in the trunk: incremental software development in the FreeBSD open source project. *Information Systems Journal*, 11, PP. 321-336.

Keil, M., and Carmel, E. (1995). Customer-Developer Links in Software Development. *Communications of the* ACM, VOL. 38, NO. 5.

Kim, A.J. (2000). *Community Building on the Web.* Peachpit Press: Berkeley, CA.

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), PP. 67-93.

Klein, H. K, and Myers, M. D. (2001). Classification Scheme for Interpretive Research in Information Systems. In Trauth, E. M (ED.). *Qualitative Research in* IS*: Issues and Trends.* Idea Group Publishing: London.

Kvale, S. (1987). Validity in the Qualitative Research Interview. *Methods: A Journal for Human Science*, 1 (2, winter): 37-72.

Lave, J., and Wenger, E. (1991). *Situated learning — legitimate peripheral participation*. Cambridge University Press: Cambridge.

Lee, A.S., Baskerville, R.L., Libeneau, J. and Myers, M.D. (1995). Judging Qualitative Research in Information Systems: Criteria for Accepting and Rejecting Manuscripts. In *Proceedings of the Sixteenth International Conference on Information Systems* (ICIS), J.L. De Gross, G. Ariav, C. Beath, R. Hoyer, and C. Kemerer (eds.), Amsterdam, December 10-13, 1995, P. 367.

Lee, G.K., and Cole, R.E. (2003). From a Firm-Based to a Community-based Model of Knowledge Creation: The Case of the Linux Kernel Development. *Organization Science*, VOL. 14(6), PP. 633-649.

Lechner, U., and Schmid, B. (2001). Communities — Business Models and System Architectures: The Blueprint of MP3.COM, Napster and Gnutella Revisited. In *Proceedings of Hawaii International Conference on System Sciences* (HICSS), Maui, January 3-6.

Lientz, B. P. and Swanson, E. B. (1980). *Software Maintenance Management*. Addison-Wesley: Reading, MA.

Ljungberg, J. (2000). Open Source as a Model for Organizing. In *Proceedings of European Conference on Information Systems*, Vienna, July 3-5, 2000.

Markus, M.L. (1983). Power, politics and MIS Implementation. *Communications of the* ACM, vol. 26(6), PP. 430-444.

Markus, M.L. (1997). The Qualitative Difference in Information Systems research and practice. In A.S. Lee, J. Liebenau, and J.I. DeGross (EDS.), *Information Systems and Qualitative Research*, PP. 11-27. Chapman & Hall: London.

McDonough, E., F., Kahn, K., B., & Barczak, G. (2001). An investigation of the use of global, virtual and collocated new product development teams. *Journal of Product Innovation Management*, 18, PP. 110-120.

Mingers, J. (1984). Subjectivism and soft systems methodology — a critique. *Journal of Applied Systems Analysis*, 11, PP. 85-103.

Mintzberg, H. (1979). An Emerging Strategy of « Direct » Research. *Administrative Science Quarterly*, Volume 24, Issue 4, Qualitative Methodology (Dec.), PP. 582-589.

Mockus, A., Fielding R., and Herbsleb, J.D. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. ACM *Transactions on Software Engineering and Methodology*, VOL. 11, NO. 3, PP. 309-346.

Moon, J.Y and Sproull, L. (2000). Essence of distributed work: the case of the Linux Kernel. *First Monday,* HTTP://WWW.FIRST MONDAY .ORG/ISSUE5-11/MOON/INDEX.HTML

Mumford, E. (1983). *Designing Human Systems*. Manchester Business School, Manchester.

Mumford, E. (1995). *Effective Requirements Analysis and Systems Design: The* ETHICS *Method*. Macmillan: Basingstoke.

Mynatt, E.D., Adler, M., Ito, and O' Day, V.L. (1997). Design for network communities. In *Proceedings for the* ACM SIGCHI *Conference on Human Factors in Computer Systems*, 1997.

Nambisan, S. (2002). Designing virtual customer environments for new product development: toward a theory. *Academy of Management Review*. VOL. 27(3), PP. 392-413.

Nambisan, S., Agarwal, R., and Tanniru, M. (1999). Organizational Mechanisms for Enhancing User Innovation in Information Technology. MIS *Quarterly*, VOL. 23, NO. 3, PP. 365-395.

Namioka, A., & Shuler, D. (1993). *Participatory Design; Principles and Practices*. Hillsdale: New Jersey.

Nandhakumar, J. and Avison, D.E. (1999). The fiction of methodological development: a field study of information systems development. *Information Technology & People*, VOL. 12 (2), PP. 176-191.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of International Conference on Software Engineering* (ICSE), 4-11 June 2000, Limerick, Ireland.

Orlikowski, W. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organizations, *Organization Science*,VOL. 3(3), PP. 398-427.

Orlikowski, W. (2002). Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science*, VOL. 13 (3).

Orlikowski, W.J. & Baroudi, J.J. Studying Information Technology in Organizations: Research Approaches and Assumptions, *Information Systems Research* (2) 1991, PP. 1-28.

Pargman, D. (2000). *Code begets community — on social and technical aspects of managing a virtual community*. Doctoral thesis, Linköping University, Sweden.

Patton, M.Q. (2002). *Qualitative Research and Evaluation Methods*. SAGE Publications: CA.

Preece, J. (2000). *Online Communities: Designing Usability, Supporting Sociability*. Chichester, UK: John Wiley & Sons.

Rheingold, H. (1994). *The virtual community: Homesteading on the electronic frontier*. Addison-Wesley; Reading, MA.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, VOL. 9, PP. 47-58.

Sawyer, S. (2001). A market-based perspective on information systems development. *Communications of the* ACM, VOL. 44(11), PP. 97-102.

Schneidewind, N (1987). The state of software maintenance, IEEE *Transactions on Software Engineering*, VOL.13, NO.3, PP.303-310.

Sharma, S., Sugumaran, V., and Rajagopalan, B. (2001). A framework for creating hybrid-open source software communities. *Information Systems Journal*, 12, PP.7-25.

Smith, M., A. and Kollock, P. (1999). *Communities in Cyberspace*. Routledge: New York.

Sommerville, I. (2001). *Software Engineering*. Addison-Wesley: Harlow, UK.

Stanoevska-Slabeva, K., and Schmid. (2001). A typology of online communities and community supporting platforms. In *Proceedings of the 34th Hawaii International Conference on Systems Sciences*.

Stake, R.E. (1978). The Case Study Method in a Social Inquiry. *Educational Researcher*, 7: 5-8.

Suchman, L. (1987). *Plans and situated actions: The problem of human-machine communications.* Cambridge, UK: Cambridge.

Tait, P. and Vessey, I. (1988). The effect of user involvement on system success: a contingency approach. MIS *Quarterly*, 1988.

Tuikka, T. and Salmela, M. (1998). WebShaman: Collaborative Virtual Prototyping in the Worl Wide Web: Supporting Internet-bases User-centered Design. In Divitini ET AL. 1998, CSCW' 98/PDC' 98 *Workshop Proceedings*, Seattle, USA.

Von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science*, 32(7), PP. 791-805.

Walsham, G. (1993). *Interpreting Information Systems in Organizations*. Chichester: John Wiley & Sons.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, VOL. 4, NO. 2, PP. 74-81.

Weick, K.E. (1979). *The Social Phsycology of organizing*. New York: McGraw Hill.

Wellman, B., and Gulia, M. (1999). Virtual communities as communities: Net surfers don't ride alone. In Smith, M. A., and Kollock, P. (EDS.) *Communities in Cyberspace*. Routledge: London, P. 167-195.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning and Identity*. Cambridge: Cambridge University Press.

Wenger, E. (2000). Communities of Practice and Social Learning Systems. *Organization*. Vol, 7(2), pp. 225-246.

Whittaker, S., Issacs, E., and O'Day, V. (1997). Widening the Net. Workshop report on the theory and practice of physical and network communities. SIGCHI *Bulletin*, 29(3), pp. 27-30.

Yin, R.K. (2003). *Case study research — design and methods*. Sage Publications: London.

Developing E-commerce in internetworked organizations: a case of customer involvement throughout the computer gaming value chain

Ola Henfridsson
Viktoria Institute • Gothenburg • Sweden

Helena Holmström
Umeå University • Umeå • Sweden

Abstract
Many computer game developers have adopted network technologies for value-adding purposes at several stages of the corporate value chain. In this paper, we suggest that this adoption extends the current notion of developing E-commerce by including ongoing interaction with the consumers concerning what is being produced. On the basis of an interpretive case study, this paper outlines the process by which a Swedish computer game developer involved its customers in producing, testing, distributing, and marketing its online computer game Clusterball. Using Orlikowski's (1999) notion of « internetworked organizations », the paper explores how this customer involvement was supported by the use of network technologies at every stage of the value chain, and it illustrates how this involvement can be understood as an important feature of future development of E-commerce in organizations.

ACM *Categories*: J.1, K.4.3, J.1, K.8.0.

Keywords: E-Commerce, Systems Development, Internetworked Organizations, Corporate Value Chain, Customer Involvement, Online Entertainment, Computer Games.

1 Introduction

Datamonitor (1999A) expects that ongoing development in both technology and customer behavior will continue to produce integration between computer game developers and Internet service providers, and that this blending will accelerate as a result of mediating more stages in the computer gaming value chain online. Meanwhile, the computer gaming industry is growing 15% annually with new standards, technologies, consumers and trends emerging, and for the involved actors — game developers, publishers, distributors, and retailers — this development requires a good portion of responsiveness towards new contingencies and opportunities.

Using Orlikowski's (1999) notion of « internetworked organizations », this paper describes and assesses one of the opportunities that E-commerce opens up for computer gaming companies, namely, the possibility to use Internet technologies to involve the customer at almost every stage of the value chain. By internetworking with the customer throughout the value-adding process, the game developer can benefit from better responsiveness to changing customer behavior and, ultimately, achieve better computer game design (CF., Ramírez, 1999). We suggest that, in putting the virtual value chain (Rayport & Sviolka, 1995) into practice, the computer gaming industry has the potential to extend the notion of electronic commerce by not only offering and distributing games online but also by involving the customer in producing them. In view of this potential extension, however, we need to know more about the process of internetworking with customers. How can companies internetwork with their customers throughout the value chain? What are the challenges involved in such internetworking?

To better understand these questions, this paper presents an interpretive case study (Walsham, 1995) of the process by which the Swedish computer game developer Daydream Software developed and launched its online game Clusterball. Clusterball represents a new generation of computer games that can be played, distributed, and paid for online. This generation of games can be considered an important step for computer developers looking for customer interaction throughout the corporate value chain.

There are several reasons why this is important. First, many researchers (see E.G., Ciborra & Failla, 2000; Timmers 1998, 1999) note that developing fruitful E-commerce is not only about systems development, but is often a matter of using technology to cultivate business processes for value-creation. Drawing on this observation, it can be useful to further understand how the customer can be part of this cultivation process by exploring an empirical case of customer involvement.

Second, throughout the history of E-commerce, the integration of various actors along the value chain has been an important issue (see E.G., Clemons & Row, 1988). Often regarded as the final stage of the corporate value chain, however, the customer's possible integration with early stages in the value chain is seldom discussed. More knowledge of such integration, therefore, can be useful for improving the development of customer-oriented E-commerce. Third, many researchers (see E.G., Normann & Ramírez, 1993, 1994; Ramírez, 1999; Wikström, 1996) present alternatives to Porter's value chain model (Porter, 1985). Many of these alternatives criticize the exclusion of the customer in Porter's model. The emergence of virtual value and supply chains (Rayport & Sviolka, 1995; Kerridge ET AL., 1998) reveals the limitations of this exclusion, and, in this sense, the research reported in this paper contributes to this discussion by illustrating how the customer of the internetworked organization can be part of almost every stage of the corporate value chain.

2 Internetworked organizations, the computer gaming value chain, and E-commerce opportunities

Diversity and conceptual innovation characterize information systems as a research discipline (Hirschheim ET AL., 1996). This diversity can be explained by the discipline's relative youth as well as by the ongoing emergence of new objects of study. Technological changes and societal development tend to call for new concepts and perspectives. In this vein, this section presents Orlikowski's (1999) internetworked organizations as a useful concept for exploring E-commerce development in interactive entertainment settings such as the computer gaming industry. Apart from Orlikowski's thinking and its related literature, this section also outlines the background to the current issues and trends of the computer gaming value chain.

2.1 INTERNETWORKED ORGANIZATIONS

In view of the increasing number of organizations that use information technology to extend business processes over traditional organizational boundaries, it has been necessary to find good labels to describe these organizations. Cybermediaries (Jin & Robey, 1999; Sarkar ET AL., 1995, 1998), imaginary organizations (Hedberg, 1991), and virtual organizations (Davidow & Malone, 1992, Mowshowitz, 1997) are three examples of such labels. All these labels are useful in that they extend the long-standing idea that organizations can be conceptualized as open systems (Lawrence & Lorsch, 1967; Scott 1992) by observing how many of today's most successful organizations use information technology in their boundary relations.

Sarkar ET AL.(1998, P. 215) refer to cybermediaries as "...*organizations that operate in electronic markets to facilitate exchanges between producers and consumers by meeting the needs of both producers and consumers*". The literature on cybermediaries highlights how the role of intermediaries is changing as a result of using network technologies and how this might require a re-formulation of transaction cost economics (Jin & Robey, 1999). The notions of imaginary and virtual organizations reveal how organizations can operate outside their traditional boundaries by using strategic alliances, information technology, and outsourcing strategies. In delivering customer value, these types of organizations build and cultivate networks with external parties rather than extend their operations outside their core competencies. In this way, these organizations can function as if they were larger than they really are, I.E., they act as imaginary or virtual organizations.

Noting how information technology nowadays is viewed as something with which to communicate, learn, collaborate, and network, Orlikowski (1999) provides an interesting exploration of how network technologies open up organizations. Such technologies enable a type of openness to various stakeholders that the traditional organization could not establish. Suppliers, legal authorities, and customers are all examples of stakeholders that are often connected to an organization's work processes and practices. This internetworking, I.E., the use of computers for connecting different stakeholders' networks, and its inter-related transparency, provide opportunities and challenges for most companies working in a global market.

In this paper, we use Orlikowski's (1999) notion of internetworked organizations for understanding customer involvement in a computer gaming setting. We find this notion particularly useful for exploring companies that almost entirely develop and maintain alliances with external parties over network technologies. This notion reveals how network technologies are increasingly interwoven in the processes of organizing. At its present stage, the research on cybermediation is useful for understanding the general nature of virtual value chains, while it offers little insight about the role of customer involvement in value-generation. In reviewing the imaginary and virtual organization perspectives, one can see how they highlight information technology as an important element in organizing, but they nevertheless attribute information technology a supportive role only. In doing so, these perspectives oversee the dual nature of information technology (Orlikowski, 1992), I.E., how information technology both enables and restricts organizational action. Network technologies do not only support the ties that hold organizations together (as assumed in mainstream virtual organization literature), but it can also be part of establishing new forms of ties such as customer involvement in product development. In this regard, the notion of internetworked organizations takes into account that network technologies increasingly provide the rules and resources for building the necessary alliances with, for instance, online customers in computer gaming.

2.2 THE COMPUTER GAMING VALUE CHAIN AND E-COMMERCE OPPORTUNITIES

In exploring the E-commerce opportunities in computer gaming, this paper uses Porter's (1985) value chain model as a starting-point for discussing the changes brought about by internetworking. This model conceptualizes the customer value of a product as a sum of added value of sequential stages. As originally formulated, these stages include inbound logistics, operations, outbound logistics, marketing and sales, and customer service, but the model is generic in the sense that the stages can easily be adapted to the particular context that it is applied to. In information systems research, the model has been used in studying, for instance, competitive advantage (Porter & Millar, 1985), inter-organizational systems (Chatfield & Bjørn-Andersen, 1997), and network technologies (Chandrashekar & Schary, 1999).

Datamonitor (1999A) expects the US and western European computer gaming market to grow around 15% annually between 1998 and 2003. Amounting to $8.7 billion in 1998, the market is predicted to be worth $17.2 billion in 2003. While this market growth signals good business opportunity, the computer gaming market continues to be sensitive to technological development, new standards, and shifting consumer behavior. In other words, as is the case in most expansive and knowledge-intensive markets, the involved actors must develop the capability of being alert to changing sources of competitive advantage (Ciborra, 1997). In their analysis of the effects of the Internet, Datamonitor (1999A,1999B) expects three points of bypass of the computer gaming value chain (consisting of game developers, publishers, distributors, and retailers). First, it is expected that console manufacturers will be able to sell upgrades as well as new games directly to consumers by offering Internet connection via their consoles. Second, PC manufacturers can be expected to deliver and sell games to consumers by having them pre-installed in the hardware. Finally, and at the center of attention in this paper, developers and publishers can be expected to bypass distributors and retailers by offering online sale of games.

In view of the changing conditions expected in the growing computer gaming market, game developers need to be alert to changing consumer behavior. A problem, however, is that there exists intermediate actors (publishers, distributors, and retailers) that can exploit their relative advantage in terms of customer knowledge achieved over a number of years. One important challenge for developers, therefore, is to find new ways to involve the customers in their business processes.

The two following sections of this paper present a case where a computer game developer intended to benefit from internetworking with its customers in its effort to develop, market, distribute, and sell its new online computer game. The case study will show how Daydream, as an example of an internetworked organization, established customer involvement throughout the value chain.

3 The Clusterball case: background and research methodology

3.1 BACKGROUND

Daydream Software is a Swedish computer game developer with its headquarters in Umeå in the northern part of Sweden. The company employs 65 people (November 2000) and the business mission is to develop entertainment that can be experienced, distributed and paid for via the Internet.

Up to July 2001, the company has developed three computer games: Safecracker, Traitors Gate, and Clusterball. Contrary to the CD-based games Safecracker and Traitors Gate, Clusterball is played, distributed and paid for online. Clusterball is an online sport in which the players fly around in ships trying to collect colored balls that are placed in different 3D landscapes. The twelve venues offer different environments (see FIGURE 1) in which the challenge is to score as many balls as possible without having them snatched by the opponents.

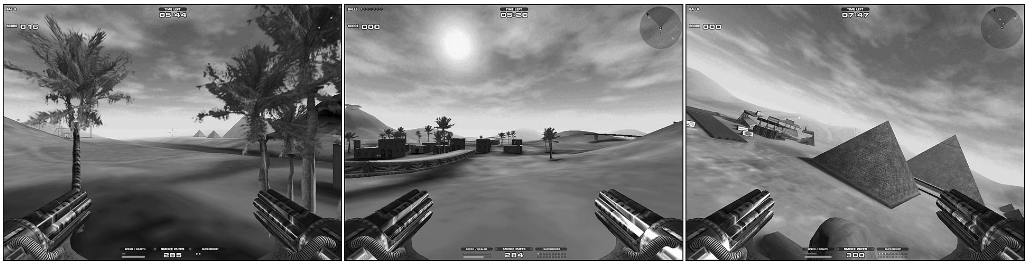


FIGURE 1. Screenshots from Egypt — one of the twelve venues in Clusterball

In releasing Clusterball (July 17, 2000), Daydream introduced a computer game with many interesting features. As many of today's most successful computer games, Clusterball is a multiplayer game that is played online. This is achieved by means of an in-house developed network protocol called Autobahn. The protocol makes possible for computer clients, independent of platform, to connect in real time for playing the game. Also, the distribution of the game is handled online. By utilizing algorithms for packet compression, packet aggregation, and latency hiding, the required bandwidth is effectively reduced so that Clusterball can be easily downloaded on a modem. Finally, the game can be paid for online. By being a content provider in the process of developing Telia PayIt — the Swedish teleoperator Telias's version of the micro-payment system Jalda — Daydream plans to offer its customers a flexible online payment system. However, while the micro-payment system is still being developed Daydream uses a traditional credit card payment system (IBM's DebiTech system).

3.2 RESEARCH STRATEGY

This study can be broadly classified as an interpretive case study (Klein & Myers, 1999; Walsham, 1995). In emerging as a valid and important approach to IS research, the interpretive approach has been widely used in understanding phenomena through the meanings people assign to them (Boland, 1985; Orlikowski & Baroudi, 1991) and the process whereby information systems influence and are influenced by a specific context (Walsham, 1993). Our study focused on the early visions, expectations and apprehensions of Clusterball and how the implementation and use of network technologies made possible for internetworking with customers.

There were two reasons for the choice of research site. First, we found Daydream as being in the forefront of using network technologies for involving customers in terms of their customer relationship management (CRM) database and virtual community (see next section). Second, we gained good access to the research site. In late 1999, the CEO introduced us to the company and pointed out the general aspects of using network technologies in the computer gaming industry. He also provided us with access to both respondents and internal data material.

To capture the informal atmosphere of the company and to get close to the research context, we conducted the study as involved researchers (Walsham, 1995). Through participant observation and by being present at the company, we aimed at getting a direct sense of the organization (Walsham, 1995) and an insider view as temporary members of the field (Van Maanen, 1979). There were at least two reasons for this approach. As involved researchers we were able to access data that normally would not be shared with outsiders. Moreover, our presence at the company enabled us to continuously discuss our interpretations with the research subjects. In doing so, we got direct response and could thereby avoid misinterpretations and misunderstandings. In accordance to the principle of interaction between the researchers and the subjects (Klein & Myers, 1999) this allowed us to critically reflect upon our interpretations and how they were constructed.

As pointed out in literature on interpretive research methodology (Walsham, 1995), there are complex researcher role issues to take into consideration when pursuing research on the basis of an intimate contact with research subjects and context. One of these issues is the problem of reporting the role of the researcher. As noted by Walsham (1995), self-reporting faces the dangers of over-modesty and self-aggrandizement by which it is hard to balance between. In coming to terms with this difficulty, we continuously documented all events we took part in during a day, in what way we considered us to influence these events and what the final outcomes of these events were. We also had research project meetings on a regular basis where we discussed our roles as researchers at Daydream. These discussions were documented and helped us in critically considering our roles as researchers at Daydream.

We were to a small part funded by Daydream for contributing with expertise in the areas of customer relationships and virtual communities. As Robey and Markus (1998) argue, practitioner sponsorship can be a valuable mechanism to ensure that the research conducted is conceived valuable. In our case, Daydream's partial sponsorship was important to get and maintain credibility in the organization as well as to gain access to key actors.

3.3 RESEARCH PROCESS

The study was conducted between January and October 2000 and can be divided into three phases. First, between January and March, an exploratory study was conducted in order to get an understanding of the different organizational actors, the every-day routines at the company, and the overall context of computer gaming. During this period, we attended company meetings and discussions, reviewed early documents on the design of Clusterball and spent time observing the practice of Daydream in order to get a notion of the daily work of the organization. During the exploratory phase of the study, data sources such as technical documents, meeting protocols and press releases were used for getting an initial understanding of the context. We also had workshops with Daydream staff and spent considerable time on informal conversations to establish a personal contact with key informants. Also, time was spent on reviewing other gaming websites to get an understanding of the global gaming community and the context in which players spend much of their time.

Second, between April and September, we conducted an in-depth study in which we spent time at the research site on a more regular basis. Complete working places were set up for the research team and, starting on April 1, we spent two or three days every week at Daydream. During this phase, we conducted 600 hours of participant observation at the company in order to complement the exploratory study with an in-depth understanding of the technologies and the design of Clusterball as well as the interpretations held and enacted by people at Daydream. While being present at the company we took part in project meetings and were part of the development of the Clusterball website by providing evaluation reports on Daydream's existing computer game websites. To be able to analyze different assumptions and intensions held by people at Daydream, notes were taken to document our conversations with employees and observations of work practice. In addition, we also documented discussions at the virtual Clusterball forum and other related gaming forums in order to capture expectations and experiences of the customers.

Third, in October, we conducted complementary data collection by maintaining close contact to what Yin (1994, P. 84) refers to as informants. These contacts were upheld by E-mail correspondence, ICQ interactions, and telephone conversations. The research process and the data sources that were used are summarized in TABLE 1.

Phase	*Data sources*	*Research goals*
January—March Exploratory study	Workshops at research site. Meeting protocols. Technical documents. Press releases. Prospects to shareholders. Informal meetings and conversations. E-mail correspondence. Official gaming websites.	Initial understanding of the research site and gaming context.
April—September In-depth study	Participant observations. Meeting protocols. Website data — logs from the official Clusterball website and logs from related fan sites. Press releases. Personal diaries. E-mail correspondence.	Identification and assessment of research questions. Interventions in the context by being temporary members of the field. Getting access to interpretations held and enacted by staff at the research site.
October Complementary data collection	E-mail. Telephone. ICQ correspondence.	Deepening the understanding of the research site. Maintaining the contact with informants.

TABLE 1. A summary of the research process and the data sources used during the study

The data analysis consisted of two intertwined processes. Firstly, we started out with two general themes — digital customer relationships and virtual communities — for exploring the case. Following the interpretive tradition in general and the principle of abstraction and generalization in particular (Klein and Myers, 1999), our data analysis consisted of an iterative process going back and forth between specifics of the implementation and use of Clusterball, and these more general themes and their implications for customer involvement throughout the value chain. Secondly, we exploited the advantage of being two researchers involved in the study. By sharing our files and continuously discussing our different interpretations of the context in which we were involved, we were able to extend our own view of the material and help each other in getting a broader perspective on specific events and statements experienced at the research site.

4 Case description: involving the customer in developing e-commerce for customer value

This section describes our understanding of Daydream's attempt to include the customer in the process of developing, testing, distributing, and marketing its online game Clusterball. The section outlines how the customer was involved throughout the value chain and, furthermore, how this customer involvement can be regarded as an important step in developing E-commerce for customer value.

4.1 DAYDREAM'S VISION

Concurring with the new stock emission in early 2000, Daydream's CEO extended the company's policy in terms of product development. *"The customer is our best product developer"*, he declared in the prospect published and distributed to current and potential shareholders, suggesting that Daydream had a lot to learn from those who play the games. This vision was based on two assumptions. First, the company's ability to respond faster to changing customer demands would increase as a result of outsourcing part of the development, evaluation, distribution, and marketing processes to those consuming the products. Second, the company's ability to respond to individual preferences would increase as a result of better communication and collaboration with individual customers.

To reach the vision about internetworking with customers, Daydream needed a technical solution that allowed Daydream to trace general changes in customer behavior and to learn about customers' individual needs. In response to these needs, a CRM database and a virtual community were developed. These can be seen as important technical components in realizing the vision about involving the customers in the development process of Clusterball.

First, the CRM database, which was developed in cooperation with a local IT-consultancy, was intended to facilitate the reach of specific customer segments. In order to do this, information about personal views and opinions had to be collected. User tests and polls, for instance, were considered as interesting means of getting an idea about the customers' favorite music, favorite TV shows, favorite style of clothes, and so on. In combination with background information such as age, gender and demographical information, it was believed that the relation to the customers could be tightened and personalized. In addition, information was collected from each gaming session in order to get information of exactly what venues individual customers preferred, how many balls he or she snatched and what opponents the player had met, ETC. By analyzing this information, Daydream would be able to learn about the customers both as private persons as well as Clusterball players.

Second, there was the Clusterball website with the virtual community. Reflecting the visions about an on-going interaction with the customers, the community was developed in order to make possible for flexible communication and functions that allowed the customers to talk to each other as well as to interact easily with people at Daydream. The website was presented in a press release dated December 17, 1999, seven months before the game was released.

> *"The new website,* CLUSTERBALL.COM, *will be the meeting place for all the Clusterball players. The goal is to build a « community » for all those who play Clusterball."* (Press release December 17, 1999)

The general wisdom of community building is that people with a common interest often generate special knowledge in that particular domain (Hagel & Armstrong, 1997; Martin, 1999). Corresponding to this, Daydream hoped that the community members themselves would be able to treat and solve many problems without interacting with the company. In this way, the community can be regarded as a multi-purpose network (Holmström & Stalder, 2001), where the community would be a place for interaction and exchange of ideas for the players while it would constitute an important resource in gathering customer-produced knowledge for Daydream.

In terms of internetworking with the customers, one can say that the Clusterball community and its related CRM database were components in the strategy of opening up the value chain to Daydream's customers and their networks of friends, game communities, and fan websites. This strategy was an important part in developing, evaluating, distributing, and marketing Clusterball, and in what follows we explore the details of this process in order to better understand what the possibilities and challenges are when developing E-commerce in internetworked organizations.

4.2 CLUSTERBALL: THE CUSTOMER'S INVOLVEMENT IN DIFFERENT STAGES OF THE VALUE CHAIN

Customer involvement in the product development stage

The traditional value chain model treats product development as one of the earliest stages in the process of building customer value, and, in practice, this stage seldom involves the customers themselves. In the Clusterball case, the involvement of customers also in product development was an important strategy for improving customer value.

However, as pleasing as this strategy was, it was not without hesitation that Daydream encouraged the players to register in the customer database. Even though this was critical in order to learn about the customers it was common knowledge that experienced players often had a negative attitude towards registration and did not voluntarily share personal information. As a consequence, Daydream decided to collect less information than they had initially planned for. Personal interests and habits are examples of information that Daydream decided to exclude in the registration process. In doing this, Daydream sought to please the community of experienced players, i.e., hardcore players, since they were seen as the most important group in the development process of Clusterball.

As stated by the CEO, the customers were seen as increasingly important in product development. Daydream had two reasons for considering customer knowledge critical in this process. First, the knowledge of the customers would be useful for product development in general. Because of the fact that customers often have lots of experience of similar products, in this case other computer games, this experience could contribute to product development in that it would better coincide with customer needs and requests. To encourage customer involvement Daydream asked the members of the Clusterball community to bring forward their opinions and suggestions on how to improve the game. After a while, many of the community-members posted their own suggestions to the virtual forum, as they knew that people at Daydream would read them. Illustrative examples are:

> *"I was sitting around today, thinking of some new play modes for Clusterball."* (« Clay » in the Clusterball community, August 28, 2000)

> *"Greatly improve the chat features for multiplayer…"* (« Shuttlekilla » in the Clusterball community, September 8, 2000)

As a result of suggestions from players, Daydream could release patches that handled configuration problems and introduced new features to the game. The first patch was released on July 18, only one day after the game had been put on the market. This was a host error patch and was a result of early testing carried out by community members. In this way, the comments from the community members were used to handle many of the most frequent problems before the majority of customers had suffered from them. The second patch was released on August 25 and included in-game screenshots, modified play modes and the ability to join the game by team name, features that had been put forward by members of the community. With the valuable comments to the second patch in mind, the developers at Daydream asked the community to make a wish list for future patches.

> *"We want your opinion on what to add to the next patch. We think that the forum will be a good place to discuss such things, so please give us your ideas."* (A posting from « Lobo » at Daydream to the gaming forum, September 7, 2000)

Besides bug fixes, the third patch included in-game chat rooms since members of the community frequently requested them and also mentioned them in the wish list they had put together.

Second, there was the idea that individual customers had individual needs and desires. Hence, Daydream strove for a product that allowed for individual design solutions. One example was the possibility to design your own « skin » (I.E., the look of the ship that you are flying in the game), a feature that was released in the second patch. The fansite BALLSNATCHERS.COM presented this feature in the following way:

> *"The upcoming patch for Clusterball will make it possible to make your own skins. Wehey! And right here on* BALLSNATCHERS.COM*, you can get the stuff you need to do so. Hell, we'll even go behind the scenes and show you what the different parts of the somewhat odd-looking figure is."* (BALLSNATCHERS.COM, August 25, 2000)

The possibility to have personally designed skins was met with enthusiasm among customers and was soon put attention to in the community. Also, experienced players included comments that would be of help to less experienced players in designing their own skins:

> *"Sometimes you want to add a scratch to your skin. And because of the not so high resolution you'll have to zoom in and work with single pixels. The picture shows how you can do it so it looks at least pretty good."* (BALLSNATCHERS.COM, August 25, 2000)

In this vein, a skin tutorial was set up at one of the related fan sites (BALLSNATCHERS.COM), so that anybody interested could get help in the design of skins. This idea reflects an important theme in the market communication that concerned a re-definition of Daydream's relationship with their customers. In introducing the open technology of Clusterball, the border between developers and customers was intended to slowly blur. In line with the open source movement (see E.G., Ljungberg, 2000; Raymond, 1999), the customers became co-designers of the game, and Clusterball was regarded an online community for entertainment designers and consumers. The core of the experience, the game, would continue to improve as more and more customers joined the community.

Customer involvement in the evaluation stage

In the traditional value chain the customer is not introduced to a product before it is thoroughly tested for potential errors. In contrast with this conventional wisdom, Daydream engaged about 200 players who voluntarily tested early versions of Clusterball.

Coinciding with the launch of the Clusterball website in December 1999, Daydream offered customers the possibility to register as « test pilots » of early versions of the game. This group of people would be valuable in detecting configuration problems and errors in the game before the game was released. During May and June 2000 the test pilots had considerable work to do. At an early stage, it became evident that there were still much to be adjusted in the game, and that the release would have to be delayed for implementing all necessary functions and meeting the requirements of different hardware configurations. In addition to the test pilots, the game was introduced at a gaming convention held in Umeå, Sweden, on July 1. During the weekend, and with Daydream people present, players from all over the country tried Clusterball and provided feedback on its features.

With the release date approaching, the efforts put on testing the game increased. On June 17, the first beta version of the game could be downloaded from the Clusterball website.

The beta version was available between June 17 and July 3, and on the Clusterball website you could read:

> *"…the purpose is primarily to locate configurations that experience troubles getting Clusterball to run. Please send us feedback on performance and any strange behavior."* (CLUSTERBALL.COM, June 17, 2000)

In addition to earlier testing that had been restricted to only a specific group of people, this announcement was an attempt to involve all potential customers in the trouble-shooting process. By voluntarily reporting errors to the virtual community, individual customers could be part of the test process before the release of the game. On June 29, Daydream's group executive officer proudly stated that "*…we are now ready to put Clusterball on the global market*", announcing that Clusterball would be available by July 17, 2000.

Customer involvement in the distribution and marketing stage

The traditional value chain model views distribution and marketing as the stages where what has been produced is packaged for meeting particular markets. This view is indeed reflected in the Clusterball case, but Daydream can be said to have extended this view by involving the players in diffusing and marketing the game among networks of gaming communities outside the direct control of the company.

Even though there existed an understanding that there was still much to be done in terms of adjusting and tuning the game, Daydream officially released Clusterball on July 17, 2000. The release was a moment of excitement and expectation in that nobody knew exactly what to expect from a computer game that represented a bypass of publishers, distributors and retailers. Daydream's bypass of these actors in the value chain can be seen as a test of Datamonitor's (1999A, 1999B) prediction that online sales of computer games would be one important opportunity for computer developers.

As a new actor in terms of distribution and marketing, Daydream had to develop strategies for taking care of these stages. In terms of marketing, the marketing department worked out a strategy on the basis of Datamonitor's (1999A) distinction between « casual » and « hardcore » players.

Casual players, I.E., people with limited experience of computer games, were considered important in that they represented a growing and prosperous market, while hardcore players, I.E., people with considerable gaming experience were important in that they had access to a network of players united in gaming communities all over the world. However, as it turned out, the choice of target group for Clusterball could not be separated from distribution and marketing considerations.

While it was desirable to reach both casual and hardcore players, Daydream decided to target one group as it was considered an effort too demanding to reach both markets simultaneously. Since it was believed that the casual players had to be reached by traditional marketing campaigns such as publicity stunts and presentations, Daydream decided to focus the marketing efforts at the hardcore players. The hardcore players could be reached easily and due to their contacts with other gaming communities they could be used in further distribution of the game. As in development and testing, it was believed that the players themselves could contribute to a great extent in the process.

To reach the group of hardcore players, Clusterball was introduced on well-known gaming websites and by personal contacts provided by people working at Daydream. In this way, the rumor about Clusterball was efficiently spread in settings in which players spent time on a regular basis.

Furthermore, the members of the Clusterball community contributed in marketing of Clusterball by publishing their own Clusterball websites. The emergence of such websites, so called fan sites, started soon after the release and in September 2000 there were already fourteen fan sites designed by individual Clusterball players (see E.G., BALLSNATCHERS.COM, CLUSTERZONE.NET, CLUSTERBALL.QUAKENEXUS.NU). The appearance of such websites proved to be important for Daydream not only in the distribution of Clusterball but in that they made possible to involve these players in future distribution and promotion of the game.

5 Internetworking with customers throughout the value chain

5.1 CLUSTERBALL AS A PLATFORM FOR E-COMMERCE

The Clusterball case illustrates how an internetworked organization such as Daydream conveys important differences to our understanding of E-commerce and the corporate value chain. In contrast with the traditional conception of the customer as a consumer of value, the internetworked organization offers the opportunity to engage customers as co-producers of value added throughout the stages of the corporate value chain. In this context, it seems that the internetworked organization's source of competitive advantage can be found in its use of network technologies to connect to various stakeholders' networks. Such network connections enable communication and collaboration with customers dispersed in terms of time and space.

There were two important components in making Clusterball useful for E-commerce. First, the Clusterball site and the virtual community were critical in that they established a technological platform for attracting a critical mass of players. As outlined earlier, this platform was important for playing, distributing, and paying for the game.

Second, a living game community was important to involve players in the different value chain stages. In light of this observation, Daydream developed certain organizational arrangements intended to support and stimulate such a community. As an illustration, a community manager was hired to cultivate the community and there were many other people in the staff who participated in this cultivation. Consider, for instance, the time and effort that Daydream staff attributed to both feeding the community with Clusterball information and actively playing the game.

In this regard, the development of Clusterball should not only be considered as computer game design, but also an attempt to establish a platform for E-commerce. But, what is new in the Clusterball case? What can we say about the nature of internetworked organizations in terms of developing E-commerce? The two following subsections highlight two issues — customer role challenges and sensemaking challenges — that characterize internetworked organizations and need to be addressed to develop E-commerce characterized by customer involvement.

5.2 CUSTOMER ROLE CHALLENGES

Internetworked organizations seem dependent on their customers. This dependency is perhaps most true in the distribution and marketing stages where the popularity and availability of an online computer game (without a CD based version sold by retailers) depends on the customers' devotion for the game. Considering that the customers' efforts in introducing the game to gaming communities by setting up fan sites, organizing clans, and much more, play a crucial role in building a critical mass of players, one understands how internetworking the distribution and marketing stages of the value chain is a delicate and critical business. While the traditional model is to set up formal agreements with distributors that market and sell the game using their established channels, internetworking these stages requires that the computer developer dare to let go of some of its control. The traditional business agreement is replaced by a type of social contract with the players.

Noting this customer dependency, the marketing department's ambivalence about deciding what information would be required from customers registering as members of the Clusterball community can be understood. The more information acquired, the better opportunities Daydream had to establish one-to-one customer relations by using the CRM database. However, this obvious advantage had to be analyzed closely against the potential damage that an integrity discussion could have had on Daydream's ability to attract hardcore players. No doubt, hardcore players, who traditionally worship integrity and condemn commercialism on the web, represent the most important group for Daydream in terms of marketing, while, on the other hand, casual players are predicted to take large part in the future computer gaming market growth (Datamonitor, 1999A). This dilemma exemplifies how internetworked organizations can be faced with the problem of approaching the customer in the role of both consumer and producer, and it also highlights how these roles often conflict with each other.

On a general level, customers have always been an important topic within the area of strategic information systems and E-commerce. On the basis of Porter's five force model (Porter, 1985), for instance, researchers have observed and documented how customers can increase or decrease their bargaining power by utilizing information technology throughout the value chain (Porter & Millar, 1985). One important difference in the case of the internetworked organization is that there are very low switching costs involved in the company—customer relation. The investment is low from both sides, and the relation is therefore more dependent on trust and devotion. In developing E-commerce for these customers, it seems that one important issue is to foster responsiveness to what generates such values. This observation also relates to research results documented in the context of information infrastructure and knowledge-intensive organization. Ciborra ET AL. (2000), for instance, show how the building of successful information infrastructures in knowledge-intensive settings requires a considerable degree of openness relative to the original plan to cater for productive exploitation of up-coming opportunities. This seems particularly relevant in business environments characterized by ongoing innovation and growth. In this regard, computer game developers have good reasons to allow drift (Ciborra, 1996) as a natural component in the E-commerce infrastructure that is developed.

5.3 SENSEMAKING CHALLENGES

Internetworked organizations seem dependent on a capacity to align customers throughout the value chain, while the common E-commerce practice is to align customers in the last stage. Observing this, one might note that there is a significant difference between the early and late stages in terms of the level of standardization that can be enforced on the producer—customer relation. Involving the customer in the early stages of the value chain, such as product development, requires a more intimate relation with the customer than a typical E-commerce standard can supply. While E-commerce has a history of, at least, twenty years of discussions about the need of common standards such as EDI and EDIFACT for enabling the exchange of messages between seller and buyer in a structured format, we know little about the needs associated with the earlier stages of the value chain.

We suggest that the early stages of the value chain call for new ways to align the customer. In fact, contrary to mainstream E-commerce, it seems that customer involvement at these stages requires a contextual approach. In a computer gaming setting, this type of understanding can be generated from and within virtual communities such as the Clusterball community. It is likely that attempts to standardize the idea-generation process characterizing virtual communities would be very unproductive in terms of the customer's contribution to product development. Instead, one might suggest that an internetworked organization should establish organizational arrangements that can transform the input that customers provide into productive and meaningful improvements of the value-generating process. Using Weick's (1979) notion of sensemaking, the most important quality of such arrangements is the ability to make sense of customer input and understand how this input relates to changes and opportunities in the surrounding business context. The active participation of Daydream staff in the Clusterball community can be seen as an embryo of handling these sensemaking challenges.

6 Conclusion

This paper presents a case study on customer involvement in producing value throughout the corporate value chain. In most E-commerce literature, the customer is viewed as the final stage of the value chain, resulting in an apparent focus on how to manage the exchange of messages between sellers and buyers in standardized formats. However, in some organizations in the computer gaming industry the customer's potential involvement in earlier stages in the value chain is tested. Using Orlikowski's (1999) notion of internetworked organizations, this paper explores how the Swedish computer game developer Daydream involved its customers throughout the value chain for producing consumer value in computer gaming.

We suggest that internetworked organizations need to develop certain alertness to shifting customer behavior and concerns. Not surprisingly when involving them as both consumers and producers, the customers' productive involvement puts the organization in a position where sudden shifts in devotion and trust can be very counter-productive. The Clusterball case provides evidence that internetworked organizations can be faced with the problem of handling the customer as both consumer and producer. This enhanced customer role is largely unexplored in the mainstream literature on E-commerce suggesting that more knowledge about this role ambiguity is needed. Increasing our understanding of this ambiguity is important for finding better strategies and models for managing the kind of product-centered E-commerce platform that Clusterball represents.

One might also observe how this ambiguity introduces sensemaking challenges for the organization. In the early stages of the value chain, the producer—consumer relation cannot be standardized, but needs a more contextual approach dominated by trust and devotion. This difference has its roots in the openness that network technologies (manifested in, E.G., virtual communities) introduce to their adopter, and it certainly needs to be handled in the process of both developing and cultivating the E-commerce components that are increasingly embedded in the practical day-to-day activity of internetworked organizations. This is an important but largely unexplored area for future research.

On a theoretical level, the internetworked organization can be considered to extend the long-standing idea that organizations can be conceptualized as open systems by providing the information infrastructure that Lawrence and Lorsch (1967) did not recognize in one of its early formulations. In those days, the use of information technology, mostly referred to as information or data processing, was seen as a vehicle for the internal handling of large amounts of information in organizations. While Ackoff (1967) was one of the earliest opponents of such processing, we suggest that today's internetworking technologies can enable the exploitation of the potential of organizations to communicate, learn, collaborate, and network with their environments including the customer. To reach this vision, however, we need leading actors like those found in the interactive entertainment business, where the lessons learned from their attempts can make us further understand how to develop and cultivate customer oriented E-commerce in the beginning of the 21st century.

Acknowledgements

The European Union's Regional Development Fund and Daydream Software funded this research. Thanks are also due to the editors of the special issue, three anonymous reviewers, Dick Eriksson, Jonny Holmström, and Rikard Lindgren for useful comments on earlier drafts of this paper. Annakarin Nyberg conducted an important part of the empirical work drawn on in this paper.

About the authors

Ola Henfridsson is the program manager of the Telematics Group at Viktoria Institute, Göteborg, Sweden. He is also an assistant professor at the Department of Informatics, Umeå University. Ola is a member of the editorial board of Scandinavian Journal of Information Systems and he has published his research in journals such as Accounting, Management, and Information Technologies, Information Systems Journal, Journal of Information and Knowledge Management, and Scandinavian Journal of Information Systems.

Helena Holmström is a PhD student in Informatics at the Center for Digital Business at Umeå University. She is also a lecturer at the Department of Informatics at Umeå University and teaches courses in systems design and electronic commerce. Her research interests focus on the use of virtual communities in software development. Helena has published and presented her research at conferences such as International Conference on Information Systems (ICIS) and IFIP WG 8.2.

References

Ackoff, R. L. (1967). Management misinformation systems. *Management Science*, VOL. 14, NO. 4, PP. B147-B156.

Boland, R. J. Jr. (1985). Phenomenology: A Preferred Approach to Research in Information Systems. In E. Mumford, R. A. Hirschheim, G. Fitzgerald, and Wood-Harper, A. T. (EDS.), *Research Methods in Information Systems*, North-Holland, Amsterdam, PP. 193-201.

Chandrashekar, A. and Schary, P. B. (1999).Toward the virtual value chain: The convergence of IT and organization. *International Journal of Logistics Management*, VOL. 10, NO. 2, PP. 27-39.

Chatfield, A. T. and Bjørn-Andersen, N. (1997).The impact of IOS-enabled business process change on business outcomes: Transformation of the value chain of Japan airlines. *Journal of Management Information Systems*, VOL. 14, NO. 1, PP. 13-40.

Ciborra, C. (1996).Introduction: What Does Groupware Mean for the Organizations Hosting It? In Ciborra, C. (ED.), *Groupware and Teamwork*, New York: John Wiley & Sons, PP. 1-19.

Ciborra, C. U. (1997). De profundis? Deconstructing the concept of strategic alignment. *Scandinavian Journal of Information Systems* VOL. 9, NO. 1, PP. 67-82.

Ciborra, C. U. and Failla, A. (2000).Infrastructure as Process: The Case of CRM in IBM. In Ciborra, C. U., Braa, K., Cordella, A., Dahlbom, B., Failla, A., Hanseth, O., Hepsø, V., Ljungberg, J., Monteiro, E., Simon, K. A. (EDS.), *From Control to Drift — The Dynamics of Corporate Information Infrastructures*, Oxford: Oxford University Press: 105-124.

Ciborra, C. U., Braa, K., Cordella, A., Dahlbom, B., Failla, A., Hanseth, O., Hespø, V., Ljungberg, J., Monteiro, E., Simon, K. A. (EDS.) (2000). *From Control to Drift — The Dynamics of Corporate Information Infrastructures*, Oxford: Oxford University Press.

Clemons, E. K. and Row, M. (1988).McKesson Drug Company: A Case Study of Economost — A Strategic Information System. *Journal of Management Information Systems*, VOL. 5, NO. 1, PP. 36-50.

Datamonitor (1999A). Electronic Games in Europe and the US to 2003, *Datamonitor* (Product Code: DMTC0315).

Datamonitor (1999B). Online Games and Gambling, 1999-2004, *Datamonitor* (Product Code: DMTC 0532).

Davidow, W. H., and Malone, M. S. (1992). *The Virtual Corporation*, New York: Harper Collins.

Hagel, J. and Armstrong, A. (1997). *Net Gain — expanding markets through virtual communities*, Boston, MA: Harvard Business School Press.

Hedberg, B. (1991). The role of information systems in imaginary organizations. In Stamper,R., Kerola,P., Lee,R., and Lyttinen,K. (EDS.), *Collaborative Work,Social Communications and Information Systems*, Amsterdam, North-Holland, PP. 1-8.

Holmström, J. and Stalder, F. (2001).Drifting technologies and multi-purpose networks: the case of the Swedish cashcard. *Information and Organization*, VOL. 11, PP. 187-206.

Hirschheim, R., Klein, H. K., Lyytinen, K. (1996). Exploring the intellectual structures of information systems development: A social action theoretic analysis. *Accounting Management &Information Technologies*, VOL. 6, NO. 1/2, PP. 1-64.

Jin, L. and Robey, D. (1999). Explaining Cybermediation: An Organizational Analysis of Electronic Retailing. *International Journal of Electronic Commerce*, VOL. 3, NO. 4, PP. 47-65.

Kerridge, S., Slade, A., Kerridge, S., Ginty, K. (1998). SUPPLYPOINT: Electronic Procurement Using Virtual Supply Chains — An overview. *International Journal of Electronic Markets*, VOL. 8. NO. 3, PP. 28-31.

Klein, H. K. and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS *Quartely*, VOL. 23, NO.1, PP. 67-93.

Lawrence, P. R. and Lorsch, J. W. (1967). *Organization and Environment: Managing Differentiation and Integration*. Boston, Graduate School of Business Administration, Harvard University.

Ljungberg, J. (2000). Open Source Movements as a Model for Organizing. European Journal of Information Systems, VOL. 9, NO. 3, PP. 208-216.

Martin, C. (1999). Net Future, New York:McGraw-Hill.

Mowshowitz, A. (1997). Virtual Organization. *Communications of the* ACM, VOL. 40, NO. 9, PP. 30-37.

Normann, R. and Ramírez, R. (1993). From value chain to value constellation: Designing interactive strategy. *Harvard Business Review*, NO. July-August, PP. 65-77.

Normann, R. and Ramírez, R. (1994). *Designing Interactive Strategy — from value chain to value constellation*. Chichester: Wiley.

Orlikowski, W. J. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, VOL. 3, NO. 3, PP. 398-427.

Orlikowski, W.J. (1999). The Truth is Not Out There: An Enacted View of the « Digital Economy ». Presented at *Understanding the Digital Economy: Data, Tools, and Research* on May 25 & 26, 1999 at the Department of Commerce in Washington, DC. (HTTP://MITPRESS.MIT.EDU/UDE.HTML, May 10, 2000)

Orlikowski, W.J. and Baroudi, J.J. (1991). Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research*, VOL. 2, NO. 1, PP. 1-28.

Porter, M. (1985). Competitive Advantage. New York: Free Press.

Porter, M. and Millar, V. E. (1985). How information gives you competitive advantage. *Harvard Business Review*, NO. July-Aug, PP. 149-160.

Ramírez, R. (1999). Value Co-production: Intellectual Origins and the Implications for Practice and Research. *Strategic Management Journal*, VOL. 20, PP. 49-65.

Raymond, E. S. (1999). *The Cathedral & the Bazaar — Musings on Linux and open source by an accidental revolutionary*. Cambridge: O'Reilly.

Rayport, J. F. and Sviolka, J. J. (1995). Exploiting the Virtual Value Chain. *Harvard Business Review*, NO. Nov-Dec, PP. 75-85.

Robey, D. and Markus, M. L. (1998). Beyond Rigor and Relevance: Producing Consumable Research about Information Systems. *Information Resources Management Journal*, VOL. 11, NO. 1, PP. 7-15.

Sarkar, M. B., Butler, B., Steinfield, C. (1995). Intermediaries and cybermediaries: A continuing role for mediating players in the electronic marketplace. *Journal of Computer-Mediated Communication*, VOL. 1, NO. 3.

(HTTP://JCMC.HUJI.AC.IL/VOL 1/ISSUE 3/SARKAR.HTML).

Sarkar, M., Butler, B. Steinfield, C. (1998). Cybermediaries in Electronic Marketspace: Toward Theory Building. *Journal of Business Research*, VOL. 41, PP. 215-221.

Scott, W. R. (1992). *Organizations — Rational, Natural, and Open Systems*. Englewood Cliffs, New Jersey, Prentice-Hall.

Timmers, P. (1998). Business Models for Electronic Markets. *International Journal of Electronic Markets*, VOL. 8, NO. 2, PP. 3-8.

Timmers, P. (1999). *Electronic Commerce — Strategies and Models for Business-to-Business Trading*, Chichester: Wiley.

Van Maanen, J. (1979). The Fact of Fiction in Organizational Ethnography. *Administrative Science Quartely*, VOL. 24, NO. 4, PP. 539-550.

Walsham, G. (1993). *Interpreting Information Systems in Organizations*, Chichester: John Wiley & Sons.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, VOL. 4, PP. 74-81.

Weick, K. E. (1979). *The Social Psychology of Organizing*, New York: McGraw-Hill.

Wikström, S. (1996). Value Creation by Company — Customer Interaction. *Journal of Marketing*, VOL. 12, PP. 359-374.

Yin, R. K. (1994). *Case Study Research: Design and Methods* (2nd ED.), Newbury Park, California: Sage.

Virtual communities as platforms for product development: an interpretive case study of customer involvement in online game development

Helena Holmström
Umeå University • Umeå • Sweden

Abstract
Information technology has changed not only the way in which we do business, but also the way in which many products and services are developed. As a structure for communication and interaction, information technology makes it possible to interweave actors such as vendors and customers in organizational processes. This paper explores how interaction in virtual communities can transform the process of product development. It does so on the basis of an interpretive case study conducted at the Swedish computer game developer Daydream Software AB. The focus of the paper is the process in which Daydream involved their customers in the development process of the online game Clusterball. By using a virtual community as a means to reach the expertise of experienced players, Daydream was able to get valuable input in the product development process. In illustrating the way in which the virtual community contributed to the development process of Clusterball, this study provides empirical support of information technology as a means to transform the process of product development.

Keywords: Virtual communities, customer involvement, product development.

1 Introduction

In a global business environment where competition is intense, there are reasons to believe that companies able to design products that are better matched to customer needs and expectations will gain competitive advantage (Ciborra and Patriotta, 1996; Shapiro and Varian, 1999). However, the ability to customize products requires knowledge about the customers. As recognized by Normann and Ramirez (1993), a key strategic task for companies is to reconfigure traditional roles and relationships within the value-creating system so that actors, such as vendors and customers, can work together in co-producing value. In this way, hidden knowledge and skills of the customers can be revealed and as a result, products that better correspond to customer expectations can be developed. Instead of regarding the customer as an object and only a consumer of value (Porter, 1980), this implies a view of the customer as a subject with knowledge that is of importance in the development of products and services (Normann and Ramirez, 1993).

As one way of involving customers in product development there is the opportunity to utilize knowledge accumulated in virtual communities. It is believed that as digital mediators or intermediaries (Chang, ET AL., 1999), virtual communities offer opportunities that can support continuous interaction between different actors such as vendors and customers.

This paper reports on research conducted at the Swedish computer game developer Daydream Software AB. The specific focus of the paper is how Daydream used a virtual community as an important component in the development and release of the online game Clusterball. Building on the idea of a community as a pool of knowledge (Hagel & Armstrong, 1997), the philosophy at Daydream was that a community consisting of experienced players would be valuable in the development process of the game.

There is considerable research about virtual communities and the impact of such as places for social interaction (Donuth, 1999; Markham, 1998; Jones, 1995; Laurel, 1993; Turkle, 1995; Preece, 2000). This body of research is important in understanding characteristics and perceptions of virtual communities as well as motivation factors and the importance of identity among virtual community members. However, there is still much to be explored in the area of virtual communities and how the expertise of such may be of value in organizational contexts. With this in mind, the attempt by Daydream to use a virtual community as a platform for product development is interesting as it illustrates the potential of virtual communities as places for on-going interaction and co-operation, as well as it represents an attempt to establish an integrated organization with focus on the customer (Raisinghani, 2000).

2 Virtual communities

The conception of virtual communities is often that of a virtual place in which people can meet to socialize, exchange experiences and enjoy the possibility to establish relationships without having to expose the physical self. There is a significant body of research conducted on how virtual worlds are conceptualized and understood (Croon Fors & Jakobsson, 2000; Markham, 1998), on virtual worlds as systems that mediate and moderate human experiences (Turkle, 1995) and how the information technology itself is a prerequisite in that it constitutes the structure within which relations can occur (Heim, 1997; Jones, 1999; Laurel, 1993; Markham, 1998; Turkle, 1995; Preece, 2000). In this respect the concept of virtual communities can be used to describe new forms of social life and the environment in which they take place. There is little doubt that virtual communities play an important role in establishing relationships between people. As recognized by Markham (1998) most people who participate in virtual environments see their interaction as real interaction with real people. This conception is conducive to the view of virtual communities a powerful arena for social interaction and unconditional relations.

However, we are now experiencing a new strand of virtual communities on the Internet. With the growing importance of the Internet for business the economic value of virtual communities has become perceptible. There are reasons to believe that companies of today are starting to realize the potential of virtual communities as a means to enable and improve customer relationships (Hagel and Armstrong, 1997), establish interaction between customers and vendors (Chang ET AL., 1999), to draw attention to their websites (Preece, 2000) and as an additional function to enhance opportunities for other business models (Timmers, 1998). Commercial virtual communities are communities with a transaction-oriented interest and where the buying and selling of particular products is of primarily interest (Chang ET AL., 1999). Hence, interaction becomes a prerequisite for the satisfaction of commercial needs.

At Daydream, the intention was to create a virtual community as a platform for product development. In contrast to traditional game development in which the customer is only to some extent involved, Daydream wanted to make possible for an on-going interaction with the customers by using the virtual community as the organizing structure for interaction. In looking back at the development process and release of the game, there were four phases in which the virtual community proved important. In what follows, these phases are presented and explored.

3 Research Design

3.1 RESEARCH METHOD

This study can be described as an interpretive case study (Klein and Myers, 1999; Walsham, 1995). For the IS researcher interested in understanding information systems in cultural and social contexts, this orientation directs the focus to people's assumptions, beliefs and desires. In the Daydream case, this meant that the early visions, expectations and apprehensions held by different actors were of importance as well as the general context of the computer gaming industry. As participant observers we aimed at getting an inside view of the work at Daydream by being temporary members of the field (Walsham, 1995; Van Maanen, 1979).

The fact that Daydream is a relatively small company in which formal decisions are rare there was a need to take part in every-day routines and assignments to capture critical information in informal speech.

The study was conducted between January 2000 and October 2000 and covers the development process of the online game Clusterball. The study can be divided into three phases. Firstly, between January and March an exploratory study was conducted. In this phase we took part in company meetings and discussions in order to get to know the employees, the organization and the every-day routines at Daydream. We also presented our research interests on virtual communities and digitally mediated relationships and the way in which we could contribute to the context by exploring these specific research questions. Secondly, between April and September we conducted an in-depth study in which we were present at the company on a regular basis, resulting in 600 hours of participant observation. During this phase we took the opportunity to bring people from different departments together in discussions related to our research interests. We also distributed reports that were part of our research so that important issues could be discussed at an early stage. Finally, in October we conducted a complementary data collection.

Two things guided our choice of research site. Firstly, Daydream represents leading practice in that the company almost exclusively uses information technology such as CRM technologies, virtual communities, a dynamic advertisement system and a micro-payment system to handle its customer relations. Secondly, the research team had very good access to the company, which is an important factor when conducting interpretive research.

3.2 DATA SOURCES AND ANALYSIS

The data sources in this study are of three different kinds. Firstly, in following the interpretive tradition, data sources such as document review and observational data were used to obtain an understanding of the assumptions and expectations held and enacted by organizational actors at Daydream. Secondly, in exploring the relationships created in the virtual community website data such as postings to the virtual forum and data collected and stored in the customer database were used. Finally, the events and meetings the researchers took part in were all documented in daily reports. During the period of the study the researchers also had E-mail accounts set up at Daydream so that we could communicate with both employees and customers.

3.3 CASE BACKGROUND

Daydream Software AB is a Swedish company originating from a company called Sombrero. Sombrero was started in 1993 and the main idea was to sell computer software and to create solutions for the graphics industry. In 1995 the owners of Sombrero joined forces with two others in setting up Daydream Software AB. The team then consisted of art directors and illustrators as well as CAD programmers and architects and the earlier focus on software development was complemented with web development and development of interactive computer games.

Today the company, consisting of 65 employees (November 2000), is developing computer games for the PC market. By the year 2000 three products have been released to the market: Safecracker, Traitors Gate and Clusterball.

With Clusterball, Daydream introduces a new generation of computer games that are distributed, played and paid for over the Internet. In relation to the game there is a virtual community designed for the players. As a virtual meeting-place the community was regarded valuable as a platform for customer involvement in the product development process.

4 The Development Process of Clusterball

4.1 VIRTUAL TEST PILOTS

In December 1999 it was made official that Daydream would put the new game Clusterball on the market in the year 2000. The game would be the first game to be distributed, played and paid for online. Also, features such as a CRM database and a virtual community made the game an interesting prospect both for the company and foreign investors. In a press release on the Daydream website it was stated that:

> *"As a first step in introducing Clusterball to the market, a website will be published on December 22. The new website,* CLUSTERBALL.COM, *will be the meeting place for all the Clusterball players. The goal is to build a « community » for all those who play Clusterball."* (Press release 99-12-17)

One of the main ideas was that the virtual community would be used in order to sort out a group of people that would be interested in initial testing of the game. By trying early versions of Clusterball, this group of people would detect technical problems such as configuration problems before the official release of the game. However, the process of starting the tests did extend in time and it was not until the early spring of 2000 that this group of people was introduced as « test pilots » of Clusterball.

During May and June 2000 the test pilots had considerable work to do. Early it became obvious that there was still much to be adjusted and that the release of the game would have to be delayed in order to meet the requirements of different hardware configurations. On June 17, the first official beta version of the game could be downloaded from the Clusterball website. The beta version would be available for anybody interested between June 17 and July 3. In this way, Daydream would get comments not only from the 200 test pilots, but from other players as well. On the Clusterball website you could read:

> *"The purpose is primarily to locate configurations that experiences troubles getting Clusterball to run. Please send us feedback on performance and any strange behavior."* (CLUSTERBALL.COM 00-06-17)

In a press release in the beginning of July it was announced that the game would be released on July 17 at 2pm. As a result of rapid feedback from community members, the first error patch was available on July 18 — only a day after the game was released to the global market. In this way, many of the most frequent problems were handled even before the majority of customers had suffered from them.

4.2 VIRTUALLY SPREAD RUMORS

While the game developers were occupied with solving technical problems, developing patches with new features and taking care of the requests put forward by the community, there were also activities going on at the marketing department. Here, the focus was how to distribute the game to the global market.

The idea was to involve players in the promotion of the game. Hence, a group of people known from the virtual community was invited to Daydream in August in order to meet with the designers, play the game and get an inside view of the company. This event was considered successful by both players and developers, and got much attention in the Clusterball community as well as in forums on related websites. In this way, the rumor about Clusterball was spread efficiently in the context in which experienced players spent time on a regular basis. Also, the members of the community contributed in the distribution of the game by publishing their own Clusterball websites. By individually designing and maintaining websites with content focusing on Clusterball, many of the players voluntarily contributed to the distribution of the game. The appearance of such websites, so called fan sites, started as soon as the game was released and in September 2000 there were already sixteen fan sites designed by individual players (see for example BALLSNATCHERS.COM, CLUSTERZONE.NET and CLUSTERBALL.QUAKENEXUS.NU). To encourage visitors to the Clusterball website to also join the discussions at the fan sites, Daydream often promoted these on the official Clusterball website:

> *"There is a new mini-game out on* BALLSNATCHERS.COM *...very quick and very fast! Head over to* BALLSNATCHERS.COM *to grab it!"* (Team Daydream at CLUSTERBALL.COM, 00-09-12)

Owing to the community members and their network of contacts the rumor about Clusterball was efficiently spread on the Internet and the game was distributed to the gaming community without much effort from Daydream.

4.3 VIRTUAL RE-DESIGN OF CLUSTERBALL

As the game was released, the virtual community and its members continued to be of importance. Foremost, the community members were involved in trying out various events that were planned as a way to attract customers. An example would be the first official Clusterball Cup that all community members were invited to join on July 27. In inviting the Clusterball community consisting of mainly experienced players, Daydream hoped for suggestions on how to improve it in order to also attract less experienced players.

Moreover, the community members were actively involved in the re-design of the game. In the virtual forum there were frequently postings concerned with future improvements of the game.

> *"I was sitting around today, thinking of some new play modes for Clusterball."* (« Clay » in the Clusterball community, 00-08-28)

> *"Greatly improve the chat features for multiplayer…"* (« Shuttlekilla » in the Clusterball community, September 8, 00-09-08)

> *"What would you think of Daydream releasing the source code so we could do some mods…"* (« BurnOut » in the Clusterball community, 00-08-27)

To encourage such postings, Daydream officially stated that they were interested in the opinions from the community members. In a posting to the forum on September 4, one of the developers at Daydream Entertainment said:

> *"At Daydream we are constantly listening to what the community wants…"* (« Lobo » in the Clusterball community, 00-09-04)

This period also included a second patch to which the community members contributed actively by suggesting what new features to add to improve the game. In this way, the virtual community was used to communicate ideas between developers at Daydream and the players during the design process.

4.4 VIRTUAL EXPRESSION OF OPINIONS

After the release, most of the discussions in the virtual community were focused around technical issues. However, as time went by discussions of a more personal character occurred. The forum came to be used to reveal personal attitudes and opinions regarding the game. An example would be the discussion in the forum on August 8, when the game had been up and running for about a month. Most players were excited but there was also the recognition of beginning problems such as few active players. In the forum you could read:

> *"It's a little quiet here, too quiet…"* (« Zodiak », 00-08-18)

As recognized by this player there was a problem in attracting new participants to the game, something that resulted in the same people challenging each other. The players paid further attention to this in the forum on August 18:

> *"I've been trying to go online to find some good games the past couple of weeks, but the only ones I seem to find are all those champs and superflys…the same people access the site every day"* (« Muerte », 00-08-18)

What followed in the community was a discussion of how to improve the game in order to attract new players. The discussion engaged lots of experienced players and together they came up with suggestions such as special tournaments and special tutorials that would help less experienced players in getting started.

These discussions were considered important to Daydream for several reasons. Firstly, they identified troublemakers and inappropriate behavior by players. This made it possible for

Daydream to respond to upcoming conflicts and solve them at an early stage. Secondly, the discussions gave a good view of the general atmosphere in the community. The postings revealed current issues that were important to the community at that particular time. Finally, the discussions in the community could help in understanding the activity in the game and what could be done to improve this. In this way, the community was seen as a tool for revealing current issues concerning the players and their overall attitude towards the product.

5 Community members as product developers

On the basis of the empirical data, this research study suggests that the virtual community was important for Daydream for involving customers in the development process of the online game Clusterball.

First, the virtual community constituted a platform for product testing. The opportunity for customers to register brought with it the possibility for Daydream to reach interested players at an early stage in the product development process. As in traditional beta testing of software products and in conformity with the open source code development (Raymond, 1999), Daydream could benefit from individual volunteers and their willingness to freely share their expertise in contributing to the development process. Bug reports and user feedback were distributed efficiently, and Daydream enjoyed the opportunity of having the tests performed directly by end users in contrast to having test cases designed and run as common in traditional system development and software engineering methods (Avison and Fitzgerald, 1995; Pressman, 2000).

To a large extent, the way in which the virtual community was used for testing resembles that of traditional beta testing of software. However, there are reasons to believe that the use of the community brought with it certain advantages. Foremost, the test process became an open process. As the bug reports were posted directly to the virtual forum the test pilots could get valuable input from each other during the test process. This made the process not only a two-way participatory activity between the company and the test participants (Greenbaum and Stuedahl, 2000), but a many-to-many communication process in which test participants could get input from each other as well as having a continuous dialogue with the developers at Daydream. Moreover, the community-building atmosphere encouraged the test participants to keep engaging in the product even after initial testing. Many of the test pilots are still devoted members of the community. For Daydream, this meant that they had customers with substantial knowledge of the product and its development history, which will bring with it the opportunity to build on accumulated customer knowledge in future product development. Finally, the ability to continuously report to the virtual forum extended the test process in time and made it an on-going process, which is rare in systems development where tests are performed during the implementation and review phases of systems development (Avison and Fitzgerald, 1995).

Second, the virtual community was used in product diffusion. In developing fan sites and by distributing the news about the product to other gaming communities on the Internet, the community members carried out the diffusion of the product in a way that has been traditionally reserved to manufacturers and large-scale distributors (Von Hippel, 2001). As a meeting-place within which relations could occur and transmit, the virtual community made possible for customer driven diffusion of the product.

Third, the community was used in product re-design. The players had concrete suggestions such as chat rooms, demo recording and playback, music for each venue and new play modes. Such suggestions were posted to the virtual forum and very early Daydream stated that all postings would be taken into consideration in the development of new patches. The community members freely shared their ideas and posted their suggestions to the community on a continuous basis. Concurring with to the open source movement (Raymond, 1999; Feller and Fitzgerald, 2000), the development process evolved incrementally as new suggestions from users appeared. However, since no source code of Clusterball was available only limited individual modifications could be made. Nevertheless, what the virtual community admitted was an open and collaborative development environment.

In view of the product development process investigated here, it is important to observe that customer involvement in software development is by no means an unknown phenomenon in the IS literature. In fact, this literature reports on several problems in involving users in systems development. As noted in user-centered collaborative approaches such as participatory design (Greenbaum and Kyng, 1991), Rapid Application Development (Avison and Fitzgerald, 1995) and Soft Systems Methodology (Checkland, 1981), one difficulty is to select user representatives that can facilitate in developing a system that match the work practice of a large user group. Moreover, it has been shown that user involvement can be hard to establish and maintain due to lack of motivation among users. In the Daydream case, the aim of customer participation was not to represent a work practice but instead to refine a product. During the process the game was used as a frame of reference to which both customers and developers could relate. Furthermore, by inviting all customers to the virtual community, Daydream got input from volunteering customers with varying skills and expertise. In this respect, participation was not imposed but customer-driven and there are reasons to believe that this encouraged both customer motivation and customer initiatives.

Finally, the virtual community was used in product evaluation. The possibility to post messages without having to expose the physical self seemed to encourage informal speech between players. These discussions gave Daydream a good view of the general conception of the product and what issues were important to the customers at that particular time. Moreover, the discussions helped in understanding the activity in the game and what could be done to improve this. Often, information about customers is generated at user sites (Von Hippel, 2001) or collected using CRM technologies (Kalakota and Robinson, 1999). However, a comprehensive view of changing customer needs calls for a considerable effort in collecting and analyzing large amounts of data. In addition to customer data obtained by using CRM technologies, the virtual community allowed Daydream to trace changing behaviors and emerging attitudes among customers by participating in the informal discussions in the virtual community. Also, much of the discussions contained information that could have been hard to capture in information requests put forward by using solely a CRM system. As an informal setting for expressing oneself, the virtual community encouraged the customers to voluntarily contribute to product evaluation by expressing their feelings for the product. Moreover, by participating in the discussions Daydream could learn about its customers in a way not easily attained by using traditional customer data collection techniques.

6 Conclusions

Looking at the development process of Clusterball, it seems that the virtual community was valuable as a platform for involving customers in the development process. By using the virtual community, Daydream succeeded in establishing a collaborative development environment dominated by interactivity, speed and continuous feedback from customers.

On the basis of empirical data, there are two implications of virtual communities in product development. First, the use of virtual communities redefines the structure of the development process. While the systems development life cycle is often described as a sequential process with a set of pre-defined phases, the virtual community enforced an integrated process with test, design and evaluation phases going on continuously. Furthermore, there was an increased possibility for customers' initiatives to direct the development process. Instead of specifying customer requirements at an early stage of the process, user requirements could be collected and implemented during the whole development process.

Second, the use of virtual communities redefines the customer role in the development process. Instead of customer involvement as a means to represent a work practice, the virtual community aimed at customer involvement in direct refinement of the product. In addition to customer involvement in the development phases of a system, the virtual community made possible for continuous customer involvement also when the system was implemented and during its subsequent use.

In this paper, virtual communities have been identified as promising platforms for product development. However, to further understand the characteristics that have been outlined and their implications, further research is needed. First, research on structure redefinition would increase our understanding of the product development process and how this might be transformed by using virtual communities as a means to coordinate an integrated development process with continuous customer involvement. Second, research on customer role redefinition would increase our understanding of how to manage product-centered virtual communities characterized by customer involvement in both the development process and after product implementation. In exploring these areas we would broaden the view of virtual communities and their potential as collaborative environments for product development.

References

Avison and Fitzgerald. (1995). *Information Systems Development: Methodologies, Techniques and Tools*. The McGraw-Hill Companies: London.

Chang A-M, and Kannan P. K., and Whinston A. B. (1999). Electronic Communities as Intermediaries: the Issues and Economics. In *Proceedings of the 32nd Hawaii International Conference on System Sciences* (HICSS), January 5-8, Maui, Hawaii.

Checkland, P. (1981). *Systems Thinking*. Systems Practice. Wiley: Chichester.

Ciborra, C. and Patriotta, G. (1996) Groupware and Teamwork in New Product Development: The Case of a Consumer Goods Multinational. In Ciborra, C. (ED.). *Groupware & Teamwork: Invisible Aid or Technical Hindrance?* John Wiley & Sons: Chichester, England.

Croon Fors, A. and Jakobsson, M. (2000). Beyond use and design — The dialectics of being in virtual worlds. *Internet Research* 1.0: The state of the interdiscipline. Lawrence, KS, USA.

Donath, J. (1999). Identity and deception in the virtual community. In Smith, M., Kollock, P. (EDS.). *Communities in Cyberspace*. Routledge: New York.

Feller, J. and Fitzgerald, B. (2000). A Framework Analysis of the Open Source Software Development Paradigm, in W. Orlikowski, P. Weill, S. Ang & H. Krcmar (EDS.) *Proceedings of the 21st Annual International Conference on Information Systems*, Brisbane, Australia, December 2000.

Greenbaum, J., and Kyng, M. (1991). Design at Work. *Cooperative design of computer systems*. Lawrence Erlbaum: Hillsdale N.J.

Greenbaum, J., and Stuedahl, D. (2000). Deadlines and Work Practices in New Media Development: Its about time. *Proceedings of the Participatory Design Conference*, New York, USA, December 2000.

Hagel, J and Armstrong, A.G. (1997). *Net Gain — Expanding markets through virtual communities*. Harvard Business School Press: Boston, Massachusetts.

Heim, M. (1997). *Virtual Realism*. Oxford University Press: Oxford.

Jones, S. (1999). Studying the Net. In Jones, S. (ED.). Doing Internet Research. *Critical Issues and methods for Examining the Net*. UK Publications: UK.

Kalakota, R., and Robinson, M. (1999). *E-Business; Roadmap for Success*. Addison-Wesley: Reading, Massachusetts.

Klein, H. K., and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, MIS *Quarterly*, VOL. 23, NR. 1, PP. 67-93.

Laurel, B. (1993). *Computers as Theatre*. Addison-Wesley Publishing Company Inc: Reading, Massachusetts.

Markham, A. N. (1998). *Life Online: Researching real experience in virtual space*. Altamira Press: London.

Normann, R. and R. Ramirez (1993). From value chain to value constellation: Designing interactive strategy. *Harvard Business Review*, July-August, PP. 65-77.

Porter, M. E. (1980). *Competitive strategy: Techniques for analyzing industries and competition*. Free Press: New York.

Preece, J. (2000). *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons: New York.

Pressman, R., S. (2000). *Software Engineering: A Practitioner's Approach*. McGraw-Hill: London.

Raisinghani, M. (2000). Electronic Commerce at the Dawn of the Third Millennium. In Rahman S., M. and Raisinghani M., S. (EDS.), *Electronic Commerce: Opportunity and Challenges*, PP. 1-20. Idea Group Publishing: London.

Raymond, E., S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly: Cambridge.

Shapiro, C., and Varian, H., R. (1999). *Information Rules*. Harvard Business School Press: Boston.

Timmers, P. (1998). Business Models for Electronic Markets. *International Journal of Electronic Markets*, VOL. 8, NR. 2, PP. 3-8.

Turkle, S. (1995). *Life on the screen. Identity in the Age of the Internet.* Simon & Schuster: New York.

Van Maanen, J. (1979). The Fact of Fiction in Organizational Ethnography. *Administrative Science Quarterly*, VOL. 24, NR. 4, PP. 539-550.

Von Hippel, E. (2001). Innovation by user communities: Learning from open-source software. MIT *Sloan Management Review*, VOL. 42, NR. 4, PP. 82-86.

Walsham, G. (1995) Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, (4), PP. 74-81.

Customer role ambiguity in community management

Helena Holmström
Umeå University • Umeå • Sweden

Ola Henfridsson
Umeå University • Umeå • Sweden
Viktoria Institute • Gothenburg • Sweden

Abstract
This paper examines challenges involved in managing product-centered communities. Using the notion of customer role ambiguity, the paper explores the ambiguity involved in balancing sound business modeling with voluntary customer participation in a computer gaming setting. The case study identifies three different customer role ambiguities — role absorption, business model violation, and non-organizational network elements — with important implications for community management. We suggest that an understanding of these implications is critical for making product-centered communities viable alternatives to traditional software development.

1 Introduction

Many of the most viable and flourishing virtual communities on the Internet are product-centered. The Linux community (Ljungberg 2000; Raymond 1999) and the Doom community (Hertz 1997) are two examples of communities that are closely associated with particular products. Many authors note how these types of communities have considerable commercial potential in terms of customer relations (Hagel and Armstrong 1997), marketing (Hoffman and Novak 1996), and product development (Wikström 1996). The underlying assumption in all these accounts is that virtual communities are useful for involving customers as co-producers in activities that traditionally are performed by companies only. In short, they portray the customer as both value consumer and producer.
While this enhanced customer role (both consumer and producer) presents companies with new opportunities, it also puts forward new organizational challenges. Needless to say, there is potential role ambiguity involved in crossing the traditional producer—customer boundary. On the one hand, customers act in the role of producers by devoting time and energy to value-adding activities such as product development and marketing without monetary compensation; on the other hand, the customers act in the role of consumers of the value produced by these activities. While there exists research on customer role ambiguity in marketing, see E.G., Beard (1999), Webb (2000), previous research on virtual communities seems to overlook this issue. How can we understand customer role ambiguity in community management?

In this paper, we identify customer role ambiguity as a critical issue for successful community management practice. Using a case study of a computer gaming community, we illustrate how customer role ambiguity can emerge in product-centered communities and, furthermore, suggest certain themes for future research on the topic.

2 Customer role ambiguity in community management: related literature

2.1 WHAT IS COMMUNITY MANAGEMENT?

In reviewing the literature, one can see that community management is broadly referred to as the activities of community development and community cultivation. Being at an early stage, the field of community management tends to be associated with a variety of aspects such as interaction and information design (Nielsen 1993, 2000; Preece 2000), technical platforms and system architectures (Lechner and Schmid 2001; Stanoevska-Slabeva and Schmid 2001) communities as business models (Timmers 1998, 1999), communities as collaboration platforms (Leevers 2001), identity-construction and trust (Donath 1999; Markham 1998; Turkle 1995), and the open source management (Ljungberg 2000; Raymond 1999). While this body of knowledge is important in understanding the characteristics and perceptions of the emerging field of virtual communities, it portrays a rather fragmented picture from which it is difficult to extract a common understanding of community management.

Throughout this paper, we refer to « community management » as the activity of establishing, maintaining, and re-producing a virtual community for commercial purposes. One distinctive feature of community management is that parties outside the traditional organizational boundary handle parts of the management activities.

2.2 CUSTOMER ROLE AMBIGUITY

Previous research on community management typically recognizes how virtual communities intersect with an enhanced customer role. Hagel and Armstrong (1996, 1997) and Martin (1999), view virtual communities as tools for creating economic value by involving customers in business processes. In other words, apart from being merely consumers of the value created, customers also produce value when engaging in virtual communities. While this opportunity is promising and especially relevant for companies aiming at the business-to-consumer market, however, the community management literature tends to overlook that this enhanced customer role is likely to come with role ambiguity.

To better understand role ambiguity, one can look closer at how this construct has been conceived in organizational theory, see Kahn ET AL.(1966) and Jackson and Schuler (1985). This literature refers to role ambiguity as an individual's uncertainty about the expectations surrounding his or her role in a job-related context (Beard 1999). One of the negative consequences of role ambiguity is that it interferes with goal accomplishment, which leads to job dissatisfaction. Even though role ambiguity has been mostly used to explain job stress and dissatisfaction, there are reasons to believe that the concept can be transferred to describe the uncertainty that customers or vendors might feel about the expectations surrounding the enhanced customer role in product-centered communities. One reason for this is that marketing researchers such as Beard (1999), Webb (2000) and Singh (1993) already have illustrated how role ambiguity can be transferred to ambiguity occurring in producer—customer relationships.

In what follows, we use a case study of a computer gaming community to illustrate how customer role ambiguity can occur in product-centered communities. These illustrations work as a basis for suggesting four themes for future research on customer role ambiguity and its implications for community management practice.

3 Illustrating customer role ambiguity: the Clusterball case study

3.1 RESEARCH SITE

Daydream Software AB is a Swedish company that develops computer games for the PC market. The company unifies competence concerning game development and interactive online entertainment. Up to 2001, the company has released three computer games:

Safecracker, Traitors Gate, and Clusterball, of which Clusterball was the first game to be played, distributed and paid for via the Internet.

In addition to technical innovations related to the online game Clusterball, Daydream also extended the company's policy in terms of product development when developing the game. In a prospect that was published and distributed to shareholders in early 2000, the CEO of Daydream declared: *"The customer is our best product developer"*. The reason to this was twofold. First, there was recognition of the value of customer expertise. The possibility to extract customer knowledge and incorporate it in new products was considered valuable for the process of product development. Second, the relationship between the company and its customers would benefit in that the two parties came closer due to a shared interest in the product. It was believed that Daydream could learn about its customers and that this would improve the company's ability to respond to general changes in customer demands and also to better meet individual preferences.

To realize the vision of involving the customers in product development, there was a need for a technical solution that enabled Daydream to interact with its customers. To serve this need a virtual community and a customer relationship management (CRM) database were developed. In the virtual community all players of Clusterball could register as members and interact with each other and with employees at Daydream in discussion forums or in a chat. In being a virtual meeting-place for all players, the virtual community provided the possibility to establish contact with a large network of players that could be of interest to Daydream. In addition, the CRM database system made it possible to learn about individual players and their preferences.

3.2 RESEARCH METHODOLOGY

This study was conducted as an interpretive case study (Klein and Myers 1999; Walsham 1995) in which assumptions and intentions held by different organizational actors at Daydream were of importance in our understanding of the context. The study was conducted between January and October 2000 and included 600 hours of participant observation in which we got to know the people and the every-day work that was carried out at Daydream.

The data sources were of three different kinds. First, data sources such as document review, official press releases and observational data were used to obtain an understanding of the company and the overall gaming context. Second, E-mail conversations, website data such as virtual message boards and data collected in the customer database were used to get an understanding of the customers and their relationship to Daydream. Finally, the events that we took part in during a day, for example company meetings and discussions were documented in personal diaries.

3.3 THE CLUSTERBALL CASE

As outlined earlier, Daydream intended to involve its customers in developing Clusterball. By introducing « Clusterball ambassadors » and by involving the customers in parts of the design work, the company enhanced the role of its customers. However, to manage this enhanced customer role brought with it certain difficulties both for Daydream and the customers.

Below, we use this case study of a computer gaming community to illustrate the enhanced customer role and how, as a result, customer role ambiguity can emerge in product-centered communities. The illustrations serve as the base for our further discussion on how customer role ambiguity might have implications for community management.

Role Ambiguity Created by Role Absorption
In order to strengthen the relationship between the company and its customers, the marketing department at Daydream introduced the concept of « Clusterball ambassadors » soon after the game was released. An ambassador would be a person with good skills in playing Clusterball, an active member of the community and a person of good language and moral that would have a positive influence on other community members. Also, an ambassador would be a person with the ability to handle conflicts without loosing the temper. According to the community manager at Daydream, there were several reasons for introducing ambassadors to the game. First, they would be able to introduce the game to their local network of contacts such as friends, people in their hometown and people in surrounding areas. Second, they would be helpful in the work of improving and realizing community activities such as administration and hosting of Clusterball tournaments. Finally, they would strengthen the relationship between Daydream and the rest of the gaming community. In doing this, the ambassadors would contribute both to daily company activities as well as in representing Daydream in its contact with potential customers.

The first ambassadors were appointed in the autumn of 2000. These persons were selected by employees at Daydream and were individually contacted by E-mail in which the community manager at Daydream explained the concept of Clusterball ambassadors. In all, five persons of different nationalities were selected and all of them were interested in becoming ambassadors in terms of what Daydream said that they expected from them.

In short, the initiative to appoint Clusterball ambassadors was successful in that it engaged players in company-related work. By the end of the year 2000 the ambassadors had hosted their own tournaments as well as contributed with content to the Clusterball website.

However, to enhance the customer role by having ambassadors as representatives of the company also brought with it customer role ambiguities. After a couple of months, there were misunderstandings between Daydream and one of the ambassadors regarding his behavior. From the company's point of view the ambassador had become too absorbed by his role as an ambassador and representative of Daydream. This was reflected in the initiatives he took in the community. Instead of being only a supportive link between the company and the players, he started to act as if he ruled the community and the community members. The ambassador had a fierce tone and commented other players in a dominant way. This behavior was considered a bad role model and the decision from Daydream was to let the ambassador leave his assignment.

According to the ambassador, one major problem was that the terms from Daydream were unclear which made him unsure of what he was expected to do on the company's behalf. Due to ambiguities in what the intentions with Clusterball ambassadors were, he found it hard to combine his role as an ordinary player with the role of an ambassador representing the company. Most often, difficulties arose when he got frustrated because of an unfair game or an irritating posting to the community forum. In such situations, he acted as if he ruled the community by giving other community members reprimands in a dominant way.

In sum, the misunderstandings between Daydream and the ambassador illustrate the problem to handle the enhanced customer role. In this case, the ambassador had difficulties in handling his role as both an ordinary player and an ambassador, which resulted in the company loosing trust in him as a company representative.

Role Ambiguity Created by Business Model Violation
Computer games are often released with a separate game engine and additional maps. According to this business model the customer is charged for the game engine but not for the additional maps that can be downloaded as patches to the game or bought as complementary CD-ROMS. For example, in the online multiplayer game Quake the customers are charged for the game engine and a predefined set of « maps » (the landscapes in which the game takes place). Additional maps and expansion packages can be downloaded and are free of charge. Moreover, the Quake players are encouraged to design their own maps, as so called « mappers ». As Quake is a successful and widespread game many other games are developed and released according to this business model. Hence, most players are used to be charged for the game engine and then enjoy complimentary features for free, as well as they are used to be part of the design work.

In opposite to this, Daydream was to profit not by selling the game engine but instead by selling Clusterball venues. On July 17 2000, the Clusterball game engine could be downloaded for free from the Clusterball website. After downloading the game engine the first two venues could be enjoyed for free. Thereafter, all additional venues had to be purchased by the customers. Another difference was that all the Clusterball venues were pre-designed by in-house developers at Daydream. Even though the players were encouraged to contribute to the design in that they could leave their suggestions on improvements in the virtual forum, employees at Daydream conducted the actual design work.

Considering the experience of most players to design parts of computer games (for example maps in Quake), it is not surprising that the Clusterball players soon wanted to have a stronger influence also over the design of Clusterball. To better meet this need, the players were offered the possibility to design their own « skins », I.E., the look of the ship that you are flying when playing Clusterball. To be able to express oneself by designing an original skin was an important feature of the game and it got much attention in the Clusterball community. However, as many games offer the possibility to design more than particular features such as skins, the gaming community soon strived for more. On August 27, there was a posting to the virtual forum in which one of the players asked people at Daydream about releasing the source code of Clusterball. This would imply that all customers would be able to modify the game and that the control of its development to a greater extent was outside the company producing it. This question was never put attention to by Daydream since a critical issue, and a prerequisite for the business model, was to make profit on the product by selling additional venues developed by in-house programmers.

In sum, this example illustrates the difficulties experienced in having customers as both consumers and producers of value. On the one hand, Daydream wanted to encourage voluntarily participation from its customers. On the other hand, there was a need to profit by commercial utilization of this participation. There are reasons to believe that this issue echoes the challenge of keeping the community members engaged as product developers at the same time as they constitute the group on which profit is to be made.

Role Ambiguity Created by Non-organizational Network Elements
Daydream intended the Clusterball community members as important components in diffusing the game. Using their networks of players, the community members were able to break the news about Clusterball to well established gaming communities on the Internet. This activity was important in the process of engaging a critical mass of players in Clusterball tournaments and team competitions. Following this engagement process, several communities outside the direct control of Daydream were established around Clusterball. On Clusterball fan sites such as ballsnatchers (www.ballsnatchers.com), clusterkings (www.geocities.com/clusterballkings) and kryptonweb (www.kryptonweb.de.vu), the most experienced Clusterball players contributed a lot in building an interest in the game by providing virtual discussion forums, reviews on gaming sessions and skin tutorials for less experienced players.

While this customer-driven activity was important to diffuse Clusterball, however, it also produced some unexpected customer role ambiguities. First, the appearance of individually developed Clusterball websites made it hard to tell where information originated. Without making it explicit to the community of players, Daydream supported certain fan sites by providing the developers with content and news about Clusterball. While Daydream argued that this made information even more credible since it was believed to originate in the gaming community itself, this made it hard for the customers to tell whether the information came from Daydream or from individual players interested in telling their own story on Clusterball. Second, the diffusion of Clusterball could no longer be fully controlled by Daydream since the activity of developing and maintaining fan sites were outside the organizational boundaries.

4 Customer role ambiguity in community management: three suggested research themes

As illustrated by the Clusterball case, the enhanced customer role in product-centered communities brings with it customer role ambiguity. In what follows, we explore how customer role ambiguity adds new dimensions to three traditional elements of business organization: trust building, business modeling, and organizational transformation.

4.1 TRUST BUILDING

As shown in research on interaction in virtual communities, see for example Rheingold (1994), Donath (1999), Turkle (1995) and Markham (1998), trust is an important property in community cultivation and a prerequisite for any virtual community to evolve in the first place. In the literature cited above, however, trust is studied among community members in non-profit communities, I.E., there is no commercial dependency between the parties involved.

Recently, trust has been recognized as one of the key factors in commercial settings such as electronic commerce and online bidding processes (Abdul-Rahman and Hailes 2000; Castelfranchi and Falcone 2000). This literature focuses on how to establish trust between trade parties that never meet physically. One of the lessons learned is that better models and representations of trust need to be developed due to different kinds and conceptions of trust (Castelfranchi and Falcone 2000; Castelfranchi and Tan 2001).

So, what are trust implications of the enhanced customer role in product-centered communities? To what extent can customer role ambiguity negatively influence trust building in product-centered communities? The enhanced customer role implies an increased dependency between vendors and consumers. When using the community to extract customer knowledge and incorporate this in products, a sudden loss of devotion and trust of participant customers would be devastating for producing value. Looking at the Clusterball case, Daydream appointed certain players as ambassadors of the game. In many respects, the cooperation between Daydream and the ambassadors worked well in that the two parties shared the similar interest of diffusing Clusterball among a larger number of players.

However, the case also illustrates how the enhanced role of the customers can be hard to manage, and how misunderstandings can occur in such relationships. Recall that one ambassador had to leave his assignment because he virtually became absorbed in his role as Clusterball ambassador. It is likely that this role absorption negatively influenced the trust built between the community and Daydream. In view of this role absorption, Daydream learnt that there are reasons to make their expectations on Clusterball customers more explicit.

The Clusterball case points out at least two important research questions that the current literature on trust and virtual communities fail to cover (see TABLE 1): In what situations can customers work as trust builders in product-centered communities? To what extent can product developers reduce customer role ambiguity by making customer expectations explicit?

4.2 BUSINESS MODELING

While it is difficult to assess and model the value generated in virtual communities, there is little doubt that successful communities do generate value for their members. This value can be, for instance, the source code in open source communities, the gathered expertise in specialized communities, or the shared pleasure of communication in MUDS.

However, the commercial setting of product-centered communities makes the value generation question more complicated. The sponsor of the community basically expects the community to generate value that can be exploited commercially. In short, this expectation creates a potential conflict of interests between the sponsor and the community members in product-centered communities, which we need to know more about.

The Daydream case illustrates that business model violation can work as a source of customer role ambiguity. Since the Clusterball business model was designed so that revenues came from selling additional venues rather than from selling the game engine, Daydream had to downplay customer involvement initiatives that were related to the venues. This downplaying occasioned ambiguity in light of Daydream's explicit intention to involve the customers in the product development process. There is little research, if any, on business model violation in relation to community management. Important research questions are (see TABLE 1): What implications does the enhanced customer role have for the design of online business models? How does one design profitable business models that still encourage voluntary customer participation?

4.3 ORGANIZATIONAL TRANSFORMATION

As noted in IS literature, see E.G., Bloomfield ET AL.(1997) and Orlikowski (1992, 1996), information technology and organizational transformation is often intertwined, and establishing successful community management is likely to require organizational transformations of different kinds.

There are recent accounts on how « internetworking » technologies such as extranets, intranets, and communication platforms make organizations more open (Orlikowski 1999). Product-centered communities are likely to contribute to such a development in that they enhance the customer role over the traditional producer-consumer boundary. Enhancing the customer role means that the distinction between the organization and its context gets indistinct. When the customer partly represents the organization, it is likely that the design of the organization needs to be adjusted in accordance with this enhanced customer role.

The Clusterball case illustrates how the enhanced customer role makes the distinction between the organization and non-organizational elements blurry. While benefiting from both the official Clusterball community and the network of non-organizational communities (ballsnatchers, clusterkings, and kryptonweb) in diffusing the Clusterball game, Daydream simultaneously lost control over the diffusion process to parties outside the organization. This lost control cuts both ways. On the one hand, it creates novel forms of intermediation. On the other hand, novel forms of intermediation might, for instance, confront existing market plans, which suggest that the sponsor of a product-centered community needs to adapt its market organization to reflect a situation where marketing is conducted both inside and outside the traditional organizational boundary. These kinds of problems are generally under-researched in the community literature. Important questions are (see TABLE 1): What are the novel forms of intermediation created by the enhanced customer role in product-centered communities? How can organizational structures and processes be designed to reflect/support the novel forms of intermediation that is created by the enhanced customer role?

Business organization elements	*Customer role ambiguities in the Daydream case*	*General Research questions*
Trust building	Role absorption	In what situations can customers work as trust builders in product-centered communities?
		To what extent can product developers reduce customer role ambiguity by making customer expectations explicit?
Business modeling	Business model violation	What implications does the enhanced customer role have for the design of online business models?
		How does one design profitable business models that still encourage voluntary customer participation?
Organizational transformation	Non-organizational network elements	What are the novel forms of intermediation created by the enhanced customer role in product-centered communities?
		How can organizational structures and processes be designed to reflect/support the novel forms of intermediation that is created by the enhanced customer role?

TABLE 1. Customer role ambiguities in community management

5 Conclusion

This paper argues that the enhanced customer role found in product-centered community settings is likely to come with customer role ambiguities such as role absorption, business model violation, and non-organizational network elements. We suggest that while role ambiguity is a critical issue for community management, it is nevertheless overlooked by current literature on this topic. In order to progress our understanding of how to successfully manage communities for commercial purposes, we therefore need to assess the consequences of customer role ambiguity for classic dimensions of business organization such as trust building, business modeling, and organizational transformation. We suggest that such an understanding is critical to make product-centered communities a viable alternative to traditional software development.

Acknowledgements

The European Union's Regional Development Fund and Daydream Software funded this study. Thanks are also due to Carsten Sørensen for comments on an earlier version of this paper. Annakarin Nyberg conducted an important part of the empirical work drawn on in this paper.

References

Abdul-Rahman, A., and Hailes S. (2000). Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences* (HICSS), January 4-7, Maui, Hawaii.

Beard, F. (1999). Client Role Ambiguity and Satisfaction in Client—Ad Agency Relationships. *Journal of Advertising Research*, March—April, PP. 69-78.

Bloomfield, B. P., R. Coombs, ET AL. (1997). *Information Technology and Organizations.* Oxford University Press: Oxford.

Castelfranchi, C., and Falcone, R. (2000). Trust Is Much More than Subjective Probability: Mental Components and Sources of Trust. In *Proceedings of the 33rd Hawaii International Conference on System Sciences* (HICSS), January 4-7, Maui, Hawaii.

Castelfranchi, C., and Tan Y-H. (2001). The Role of Trust and Deception in Virtual Societies. In *Proceedings of the 34th Hawaii International Conference on System Sciences* (HICSS), January 3-6, Maui, Hawaii.

Donath, J. (1999). Identity and deception in the virtual community. In Smith, M., Kollock, P. (EDS.). *Communities in Cyberspace*. Routledge: New York.

Hagel, J., and Armstrong, A., G. (1996). The Real Value of Online Communities. *Harvard Business Review*, May—June, PP. 134-141.

Hagel, J and Armstrong, A., G. (1997). *Net Gain — Expanding markets through virtual communities*. Harvard Business School Press: Boston, Massachusetts.

Herz, J. C. (1997). *Joystick Nation*. Abacus: London.

Hoffman, D. L. and Novak, T. P. (1996). Marketing in Hypermedia Computer-Mediated Environments: Conceptual Foundations. *Journal of Marketing*, 60, July, PP. 50-68.

Jackson, S., E., and Schuler, R., S. (1985). A Meta-Analysis and Conceptual Critique of Research on Role Ambiguity and Role Conflict in Work Settings. *Organizational Behavior and Human Decision Processes* 36(1), PP. 16-78.

Kahn, R., L, Donald, M., Wolfe, R., P., Quinn, J., Diedrick, S., and Rosenthal, R., A. (1966). *Organizational Stress: Studies in Role Conflict and Ambiguity*. Wiley: New York.

Klein, H. K., and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS *Quartely* (23:1), PP. 67-93.

Leevers, D. (2001). Collaboration and Shared Virtual Environments — from Metaphor to Reality. In Earnshaw, R., Guedj, A., and Vince, J (EDS.). *Frontiers of Human-Centered Computing, Online Communities and Virtual Environments*, PP. 278-299. Springer: London.

Lechner, U., and Schmid, B. (2001). Communities — Business Models and System Architectures: The Blueprint of MP3.COM, Napster and Gnutella Revisited. In *Proceedings of the 34th Hawaii International Conference on System Sciences* (HICSS), January 3-6, Maui, Hawaii.

Ljungberg, J. (2000). Open Source Movements as a Model for Organizing. *European Journal of Information Systems* 9(3), PP. 208-216.

Markham, A. N. (1998). *Life Online: Researching real experience in virtual space*. Altamira Press: London.

Martin, C. (1999). *Net Future*. McGraw-Hill: New York.

Nielsen, J. (1993). *Usability Engineering*. AP Professional: Boston, Massachusetts.

Nielsen, J. (2000). *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing: Indianapolis, IN.

Orlikowski, W. J. (1992). The duality of technology: Rethinking the concept of technology in organizations. *Organization Science* 3(3), PP. 398-427.

Orlikowski, W. J. (1996). Improvising Organizational Transformation Over Time: A Situated Change Perspective. *Information Systems Research* 7(1), PP. 63-92.

Orlikowski (1999). The Truth is Not Out There: An Enacted View of the « Digital Economy ». Presented at *Understanding the Digital Economy: Data, Tools, and Research*, on May 25-26 at the Department of Commerce in Washington, DC. (HTTP://MITPRESS.MIT.EDU/UDE.HTML)

Preece, J. (2000). *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons: New York.

Rheingold, H. (1994). *The Virtual Community. Finding Connection in a Computerized World*. Secker & Warburg: London.

Raymond, E. S. (1999). *The Cathedral & the Bazaar — Musings on Linux and open source by an accidental revolutionary*. O'Reilly: Beijing.

Singh, J. (1993). Boundary Role Ambiguity: Facets, Determinants and Impacts. *Journal of Marketing*, 57(2), PP. 11-31.

Stanoevska-Slabeva, K, and Schmid, B. (2001). A Typology of Online Communities and Community Supporting Platforms. In *Proceedings of the 34th Hawaii International Conference on System Sciences* (HICSS), January 3-6, Maui, Hawaii.

Timmers, P. (1998). Business Models for Electronic Markets. *International Journal of Electronic Markets* 8(2), PP. 3-8.

Timmers, P. (1999). *Electronic Commerce — Strategies and Models for Business-to-Business Trading*. Chichester: Wiley.

Turkle, S. (1995). *Life on the screen. Identity in the Age of the Internet*. Simon & Schuster: New York.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*. 4, PP. 74-81.

Webb, D. (2000). Understanding Customer Role and its Importance in the Formation of Service Quality Expectations. *The Service Industries Journal*, January, 20(1), PP.1-21, 2000.

Wikström, S. (1996). Value Creation by Company — Customer Interaction. *Journal of Marketing*, VOL. 12, PP. 359-374.

The distributed nature of software development — a comparison of three development approaches

Helena Holmström
Viktoria Institute • Gothenburg • Sweden

Abstract
Much change has undergone the environment in which software development takes place. To a greater extent, we are experiencing a distributed development environment. In this paper, three approaches to distributed software development are identified and explored — global software development (GSD), open source software development (OSS) and community-based software development (CSD). In a comparison of these, it is argued that the approaches embrace differences that are important to take into consideration for companies entering the distributed environment of software development. This paper suggests that these differences are related to the dimensions of (1) nature of development approach, (2) communication structure, and, (3) coordination mechanisms.

Keywords: Distributed software development, distributed development approaches

1 Introduction

Much change has undergone the environment in which software development takes place (Sheremata, 2002; Orlikowski, 2002; Feller & Fitzgerald, 2002). Today, software developers work under increasing competitive pressures and the systems themselves are continuing to be more technically advanced. Also, there are changing customer requirements to take into consideration and an accelerating demand for more customizable features in the software that is produced. There is little doubt that these aspects make software development an unwieldy process.

However, what really strike you are the changes in the way the software development process is coordinated. To a greater extent, software development is becoming a distributed process and there is the need for development approaches that take into consideration the distributed environment in which developers and users communicate and coordinate their work.

In this paper, three approaches to distributed software development are identified and explored. First, there is global software development (Carmel & Agarwal, 2001; Dubé & Paré, 2001; McDonough ET AL., 2001). Second, there is open source software development (Feller & Fitzgerald, 2002; Raymond, 1999; Gallivan, 2001; Sharma ET AL., 2002; Bergquist & Ljungberg, 2001). Third, there is community-based software development (Holmström, 2001).

To illustrate the different approaches, three empirical cases are presented. In a comparison of these, it is argued that the approaches embrace differences in terms of: (1) nature of development approach, (2) communication structure, and, (3) coordination mechanisms. To this end, this paper aims at creating an understanding of the different approaches to distributed software development that can be taken, and in this way, facilitate for software development companies entering the distributed development environment.

2 Background

Below, three approaches to distributed software development are identified — global software development (GSD), open source software development (OSS) and community-based software development (CSD). In the following discussion, these different approaches constitute the background for a comparison of three empirical cases.

2.1 GLOBAL SOFTWARE DEVELOPMENT

In global software development (GSD), geographically distributed, and culturally diverse, software developers or development teams work jointly in a software development project (McDonough ET AL., 2001). According to Carmel and Agarwal (2001), there are at least 50 nations participating in collaborative software development internationally, and the number is rapidly increasing due to economical benefits of outsourcing, the growth of the global market and the occurrence of business arrangements such as strategic partnerships and joint ventures (Karolak, 1998). To support this type of distributed development there is the need for communication and cooperation technologies. As recognized by Smith and Blanck (2002), a combination of technologies can be used where synchronous media such as videoconference systems, electronic meeting systems and virtual whiteboards are combined with asynchronous media such as voicemail, electronic bulletin boards and forums, E-mail and group calendars.

Global software development is not characterized of user-driven development. Instead, the software is developed by professional developers and then sold to users around the world. In this sense, the users can be seen as consumers of the value that is produced. Hence, the key challenge during software development is not on techniques for user-developer interaction, but rather on techniques to support developer—developer communication and coordination.

2.2 OPEN SOURCE SOFTWARE DEVELOPMENT

In conformity with global software development in which software development is performed by geographically distributed developers, there is open source software (OSS) development. Recently, this movement has gained significant attention and it is believed that OSS development has the potential to influence the future of organizations both in terms of organization, customer relations and business models (Ljungberg, 2000).

In OSS, communities of developers contribute on a voluntary basis in developing software that is freely shared for review, reuse and modification. As recognized by Ljungberg (2000), the work seems to be totally distributed, delegated and loosely coupled. In terms of organization, the OSS development approach does not have any formal structure (Sharma ET AL., 2002). Hence, projects are not dictated by any formal schedule or list of deliverables, neither is work assigned to the developers. No particular development method is advocated and unlike conventional software development there is no formal procedure to ensure that developers are not duplicating effort by working on the same problem at the same time. On the contrary, this is seen as beneficial to the process since it allows for a competition among multiple high-quality solutions (Feller & Fitzgerald, 2002). To coordinate the process, configuration tools such as the Concurrent Versions System (CVS) are used. The CVS offers an easy way to incorporate changes to the repository and with one single command the developers can download the latest version of the software tree (Feller & Fitzgerald, 2002). Lately, the CVS has also been complemented with web based extensions such as Bugzilla (web based bug tracking), Bonsai (web based access to archived source code) and Tinderbox (web based tools for analysing software builds).

In contrast to global software development, OSS development is characterized by a complex definition of who is the user and who is the developer. Often, the developer and the user of one particular piece of code is the same person. According to Ljungberg (2000), this is due to the fact that most OSS projects originate in individual needs and requirements.

Recently, there have been studies highlighting the importance of virtual communities as platforms for software development (Holmström, 2001; Henfridsson and Holmström, 2002). In particular, OSS literature has contributed to the view of virtual communities as enablers for collaborative work between distributed people (Sharma ET AL., 2002; Scacchi, 2002). As recognized by Sharma ET AL.(2002), OSS development is a fundamentally new way to develop software and the large number of voluntary developers reflects the strength of the community culture and the « sense of community » that can be found in virtual groups of like-minded people (Blanchard and Markus, 2002).

However, while there are significant benefits of OSS development, there is not always the possibility for software companies to deploy this way of distributed development. Most often, for-profit organizations have difficulties in building business models around the OSS paradigm (Sharma ET AL., 2002). Instead, there is the challenge of finding ways to incorporate aspects of the community culture into traditional software development processes — and in this way to allow for a community-based approach to software development in which user involvement and user participation are key characteristics.

To do this, there is the possibility to create « hybrid communities » (Sharma ET AL., 2002). In these, features found in OSS communities are infused to varying degrees into traditional organizational structures to facilitate for flexible and user-driven development of quality software. Evidence points to leading organizations like Hewlett Packard, IBM, Intel, Sun Microsystems, ETC., already having taken steps to use communities as a way to incorporate elements of OSS into their software development processes (Sharma ET AL., 2002). This indicates the belief in communities as valuable for involving users in the development process. Also, the benefits of (A) reduced development time, (B) improved quality, (C) reduced cost, (D) gained developer loyalty, and (E) increased developer talent pool can be enjoyed (Sharma ET AL., 2002).

In resemblance with the hybrid communities presented by Sharma ET AL.(2002), community-based software development requires the consideration of three major elements: (A) community building, (B) community governance, and (C) community infrastructure. First, « community building » refers to the precondition of having a « community of practice » (Wenger, 1998) with a strong and shared culture. To support this, organizations need to provide a free flow of information, to get rid of the formal organizational structures and provide mechanisms for informal relationships and networking among community members (Sharma ET AL., 2002). Second, « community governance » refers to the implementation of transparent governance mechanisms. Here, managers have to move away from the practice of imposing central command and control and, instead allow for community members to work in teams and to make decisions by discussing and voting (Sharma ET AL., 2002). Third, « community infrastructure » refers to the tools and infrastructures necessary for software development. In resemblance with the CVS-system that is used within OSS development (Feller and Fitzgerald, 2002), community-based software development presupposes a central repository in which information is accessible for the community members (Sharma ET AL., 2002).

Based on this background, three empirical cases of distributed software development are presented. In the following discussion, the three cases are compared in terms of (1) nature of development approach, (2) communication structure, and (3) coordination mechanisms.

3 Research setting and method

The empirical part of this paper is based on three case studies. To obtain this data, two different methods for data collection were used. First, a secondary analysis of two published case studies on distributed software development was employed. Second, an interpretive case study at a software company was conducted.

First, the secondary analysis implied the identification of representative papers. To identify the first case study — the case on global software development — I used the search engine Google and the scientific literature digital library CiteSeer which uses the search engines AltaVista, HotBot, and Excite to identify publications within the field of computer science and information systems. In this search, I used terms such as « global software development », « distributed software development » and « distributed development teams » to identify studies within this area. Although many publications were available on global software development, there were few case studies in which original data and data analysis could be obtained. Finally, by backtracking a reference found in a paper on product development, a study by Orlikowski (2002) was identified. In being an empirical account of the work conducted in a geographically dispersed organization this study matched the search criteria and was selected to represent the global software development approach.

The second case study on which secondary analysis was employed was identified in the reading of Information Systems Journal and the special issue on Open Source (ISJ, nr 11, 2001). In one of the papers, Gallivan (2001) identifies nine case studies of OSS development. According to Gallivan, these studies were identified after searching the electronic archives of both ACM (The Association for Computing Machinery) and IEEE (Institute of Electronics and Electrical Engineers), after searching the database Bell & Howell/Proquest's AB/Inform and after reviewing the papers that were presented at the 1st Workshop on OSS Engineering (Feller ET AL., 2001). From the hundreds of publications that were found, only nine fit the selection criteria outlined by Gallivan (2001). First, the publication needed to describe the process of OSS development, in general, or one or more specific OSS projects. Secondly, the paper had to contain original data and analysis. From the nine case studies that were identified by Gallivan (2001), the Moon and Sproull (2000) study was selected to represent the OSS development approach described in this paper.

The third case study presented in this paper is the study of community-based software development. The empirical work reported here builds on an interpretive case study (Walsham, 1995; Klein & Myers, 1999), conducted by me and a research colleague at

Daydream Software between January 2000 — October 2002 (for publications on this study see for example, Holmström, 2001; Nyberg and Henfridsson, 2001; Henfridsson and Holmström, 2002). In our study, we focused on the software development process of the online game Clusterball and the way in which a virtual community was used to involve distributed users in the development process. In the study, data sources such as technical documents, meeting protocols, press releases and printouts from the community forum were used. Also, an extensive review of other gaming websites was conducted. Furthermore, the specific context of Daydream and its customers was explored through 600 hours of participant observations at the company, 14 qualitative interviews with Daydream employees and a web survey that was sent out to 200 community members.

4 Distributed software development: three empirical cases

In section two, three approaches to distributed software development were identified. Below, these approaches are explored in three empirical cases. First, there is the Kappa case, representing global software development (Orlikowski, 2002). Second, there is the Linux case, representing OSS development (Moon and Sproull, 2000). Finally, there is the Clusterball case, representing community-based software development (Holmström, 2001). While the first two cases are based on secondary analysis of published case studies, the third case is based on an interpretive case study conducted by the author.

4.1 GLOBAL SOFTWARE DEVELOPMENT — THE KAPPA CASE

Kappa is a globally-dispersed software development company with its headquarters in The Netherlands. At Kappa, the software development efforts are accomplished through temporary, global project groups involving a few hundred software engineers from all over the world. The software development activities are distributed across multiple Development Units (DUs) located in 15 different locations spread over 5 continents. The geographical and cultural diversity of Kappa can be understood in the following statement made by one of the managers:

> *"My situation is quite typical…I am a Greek working in Finland for a Dutch company and using English to do my work."*

The rationale for having these distributed development teams is both economical and strategical. While the economical aspects include an increasing competitive pressure to reduce the time-to-market and an accelerating demand for more customizable features in the software, the strategical aspects are touched upon by one of the senior executives:

> *"…First of all, you get access to resources wherever it is. Holland is a pretty small country and our universities just don't turn out the number of engineers that Kappa needs…Another advantage is proximity to the markets."*

> *To manage the complexity of the distributed work at Kappa there is a well-established organizational structure. In each location there are senior executives, senior* DU

managers, project managers, subproject managers and software engineers. However, as observed by one of the senior executives, there are difficulties in being such a diverse organization:

"… as much as it is very nice to have these organizations that are diverse, they also sometimes pull in different directions. And the big challenge is to bring them together."

To handle negotiations and discussions there are substantial aligning efforts within Kappa. These are accomplished through two key activities: (1) the use of a proprietary project management model, its planning tool and structured systems development methodology, and (2) the annual contracting for work via standard metrics. A senior executive commented on the role of project planning and methodology tools in facilitating distributed work:

"We use a common process methodology…And then we have coordination within this framework, done at all levels of the project to get all the different software pieces together for the system at the same time. There are the technical standards and coordination documents…."

As part of the project plan, several documents have to be written. For example, there is an operational plan, an assignment specification and a project specification. The importance of Kappa's project management model and methodology in aligning the different projects is explained by one of the project managers:

"The project model and methodology helps a lot…We develop requirement specifications, development sketches, implementation proposals, technical reports, everything that tells us at an early stage, this is the scope, this is feasible, this is what we are going to do and this is what it costs now."

The use of the project management model as well as the division of projects into subprojects is all part of the hierarchical decomposition of work that characterizes Kappa. However, despite its efficiency of coordinating the different projects, the proprietary suite of the project management model can be viewed as constraining in shifting to new software platforms, new infrastructures, new programming languages and new development methodologies. One of the managers commented about Kappa's current project management model:

"I think it helps us, but the drawback is that the limit has been hit now of the capacity of that model…what we need now is a new model and a new methodology for parallel development."

Also, Kappa members emphasize the importance of face-to-face meetings. Despite the qualities of the project management model, these are necessary for establishing social relationships. One project manager noted:

"You can't resolve everything over the phone. It is important to have that personal relationship as well, which you achieve by meeting each other, and then it makes it a lot easier when you communicate through E-*mail or the phone."*

The statements above reflect both advantages as well as limitations in working in a distributed environment. At Kappa, 30 nationalities in 15 geographic locations are currently trying to align their software development efforts. Their customers, on the other hand, seldom experience anything but a final product. Unaware of the distributed nature of Kappa they turn to their local subsidiary when having any problem.

4.2 OPEN SOURCE SOFTWARE DEVELOPMENT — THE LINUX CASE

In the beginning, Linux was a PC-based operating system produced through a software development effort consisting of more than 3,000 developers distributed over 90 countries on five continents. In its first three and a half years of development more than 15,000 people submitted code or comments to the three main Linux related newsgroups and mailing lists. As of December 1998, more than eight million users were running Linux on a wide variety of platforms and the operating system was projected to have an annual growth rate of 25% (Shankland, 1998). Today, Linux is much more than an operating system. As the number of people interested in Linux grew, they formed user groups to share information through the Internet with any Linux user in the world. By July 2000, there were more than 400 Linux user groups in 71 countries.

The real fascination with Linux stems from the fact that it is not an organizational project. Instead, volunteers from all over the world contribute code, documentation and technical support because they want to. The first posting regarding the project came on August 25, 1991, when a computer-science student from Helsinki wrote:

"Hi everybody out there using « minix » — I'm doing a (free) operating system (just a hobby, won't be big and professional like GNU*) for* 386 (486) AT *clones. This has been brewing since April, and is starting to get ready. I'd like any feedback on things people like/dislike in « minix » ...."*

This was followed by the announcement of Linux V0.02 on October 5, 1991. In a message posted to one of the newsgroups on the Internet, Linus Torvalds — the Helsinki student — wrote:

"This is a program for hackers by a hacker. I've enjoyed doing it, and somebody might enjoy looking at it and even modifying it for their own needs...and I'm looking forward to any comments you might have."

Furthermore, everybody interested was invited to join the project:

*"Are you without a nice project to modify for your needs? Then this post might be just for you...Full kernel source is provided, as no « minix » code has been used...Sources to the binaries (*BASH *and* GCC*) can be found at the same place in* /PUB/GNU.*"*

Until the year 2000, there were 569 additional releases, all managed and announced by Linus Torvalds who single-handed acts as a filter on all patches and new releases. Depending on his judgment, a contribution can be rejected, accepted or revised. However, to help him in his decisions and in his programming efforts, Linus has an active community of programmers, who, electronically organized, are crucial for advice, suggestions and code. By using Linux mailing lists and Usenet groups, Linux community members get continuously updated on where to send code and where to find information.

In the Linux community, the role structure has been identified as important for the overall organization of the development work. The two most important roles are « credited developer » and « credited maintainer ». The credited developer role originates from the v1.0 release in 1994 in which a credits file was included to publicly acknowledge people who hade contributed substantial code to the kernel. The credited maintainer role was formally acknowledged in February 1996 when the maintainers file was announced. Designated maintainers are responsible for particular modules of the kernel, for example, they review Linux-kernel mailing list submissions relevant to their modules, build them into larger patches, and submit the larger patches back to the list and to Linus directly.

4.3 COMMUNITY-BASED SOFTWARE DEVELOPMENT — THE CLUSTERBALL CASE

Daydream Software is a Swedish game developer. During the time for the study, the company had 65 employees ranging from software developers, graphical designers and web designers to marketing people, administrators and managers. At Daydream, the developers are located in offices close to each other. They communicate face-to-face or by using the telephone, but no major efforts are needed to support their communication and coordination electronically. Instead, the nature of distribution lies in the intention by Daydream to involve its distributed users in the software development process, an intention that was announced by the manager in 1999 in relation to the development of the online game Clusterball:

> *"Our customers are our best product developers. We want constant feedback on our design suggestions so that we know what they want and how they want the product to improve. We want them as part of the design process."*

While the developers are co-located, the customers are distributed around the world. This posed several challenges to Daydream in terms of communication and coordination tools. To solve this, a virtual community was created in which there was the possibility for users and developers to communicate using electronic forums, chats and E-mail. Very soon, the developers used the community forum to encourage user feedback on one of the beta versions of Clusterball:

> *"The purpose is primarily to locate configurations that experiences troubles getting Clusterball to run. Please send us feedback on performance and any strange behaviour."*

The response from the community was positive and as soon as the beta version was made available for downloading, the feedback could be enjoyed:

"I would like to see the ability to set a minimum and maximum player ranking when I host a game. In this way, a « Newbie game » will really be for « Newbies » — experts won't come along and thrash everyone."

"The number one thing I would like to see is demo-recording and playback".

The postings revealed configuration problems, modifications and future suggestions on additional functionality. Only between July 17, 2000 (official release date) and November 2002 (the end of the study) there were 15,667 postings to the general forum and 1,878 postings to the technical forum. Realizing this, Daydream expanded its organization and appointed a « community manager » to handle the postings and the activity in the forum. Also, the community manager was responsible for communicating the ideas put forward by the community to the rest of the company. According to one of the community members, their influence on the product was significant:

"I think peoples' suggestions on new features to patches are definitely taken into consideration…the people at Daydream seem to be open to suggestions from us players. "

Also, the developers seemed to enjoy the interaction with community members. In the following statement one of the developers reflects on the benefit of having community members influencing the product:

"I use the community a lot in my work. In reading the postings I always find good suggestions on what to improve. Also, it is fun — I feel like I learn about the customers and what they really want!"

5 Discussion

As illustrated in the empirical cases, there are different approaches to distributed software development that can be taken. Below (TABLE 1), the approaches are compared in terms of three dimensions: (1) nature of development approach, (2) communication structure, and, (3) coordination mechanisms.

As can be seen, there are significant differences in the nature of development approach. In GSD, there are distributed teams of developers motivated by strategical and economical rationales. In OSS, individual developers contribute without monetary compensation in software projects where the code is freely shared due to altruistic values. Often, projects originate in individual needs, something that is evident in the Linux case where Linus Torvald's own objective was to create a Unix-like operating system for the IBM PC 386 series (Feller and Fitzgerald, 2002). In CSD, software is produced by developers in close cooperation with distributed individual users in resemblance with the idea of user participation as expressed in for example Participatory Design (Namioka and Shuler, 1993) and Contextual Design (Beyer and Holtzblatt, 1998). These differences are further visible in

Dimensions	*Global Software Development (GSD)*	*Open Source Software Development (OSS)*	*Community-based Software Development (CSD)*
(1) NATURE OF DEVELOPMENT APPROACH			
Design rationale	Strategical/Economical	Altruism/Ideology	User participation
Developer infrastructure	Distributed teams of developers	Distributed individual developers	Distributed individual users
Role of developer	Producer of software	Producer/consumer of software	Producer of software
Role of user	Consumer of software	Consumer/producer[4] of software	Consumer/producer[5] of software
Division of work	Hierarchical decomposition of work	Parallel development	User—developer iteration
(2) COMMUNICATION STRUCTURE			
Communication type	Developer—developer	Developer—developer Developer—user User—user	Developer—user User—user
Communicative actors	Project managers Senior executives Senior managers Sub-project managers Project members	Credited developers Credited maintainers Community members	Project managers Community managers Community members
(3) COORDINATION MECHANISMS			
Coordination infrastructure	Face-to-face, phone & audio conference, videoconference, electronic meeting systems, virtual whiteboards, data conferencing , voicemail, mail, electronic forums, intranets, E-mail, group calendars	Configuration management systems, web based bug tracking systems, web based access to source code, web based tools for analysis, virtual community functions such as electronic forums, news groups, E-mail	Virtual community functions such as electronic forums, chats, E-mail
Coordination tools	Project management models Development methods	Peer supervision Peer review	Project management models Development methods Community postings
Coordination process activities	Requirement specification Implementation proposals Software design Software test and implementation Software maintenance	Problem discovery Solution identification Code development and review Code commit and documentation Code release	Software design Software release Community-driven test and review Community-driven modification and maintenance

TABLE 1. Comparison of three approaches to distributed software development

[4] In terms of software code that can be implemented.
[5] In terms of feedback, design suggestions and modifications of existing design.

the roles of developers and users. While GSD and CSD developers are mainly producers of software, the OSS approach is characterized by developers also being users of the software. Furthermore, OSS, and to some extent CSD, allow for parallel development, something that is not encouraged in GSD.

Due to the diversity in development infrastructure, there are differences in the communication structure. While the GSD approach focuses on developer—developer communication, both OSS and CSD support communication also between developers and users as well as between users and users. Regarding the communicative actors, there is evidence of hierarchical structures in all three approaches. This is interesting, especially in relation to the OSS and CSD approaches. While the community culture is often described as a bazaar (Raymond, 1999) where chaotic development processes evolve into coordinated processes, the cases presented in this paper bear evidence of defined structures and hierarchical organizations also within these. In the Linux case, Linus Torvalds dictated the rules, and in the Clusterball case, a community manager was appointed to direct the community activities.

The infrastructure and the tools that are necessary to align distributed developers and distributed users are included in the final dimension of coordination mechanisms. In all approaches, Internet technology is deployed as the infrastructure for coordination. However, while the OSS and CSD approaches depend solely on Internet-based coordination infrastructures, the GSD approach encourages physical interaction and face-to-face meetings. In these, project management models and other formal structures can be negotiated, an activity that is not evident in for example OSS development communities. Rather, these are self-organized and self-governed. Furthermore, there is an interesting difference in the coordination process activities that are carried out in each approach. While the design evolves as a result of an iterative process in both OSS and CSD, the GSD approach advocates for a pre-defined process in which requirement specifications direct the process. This is evident in the Kappa case were project specifications were an important part of the project management model.

To sum up, the approaches embrace different characteristics important to distributed software development. While the GSD approach is emerging due to industry and business drivers on the global market (Karolak, 1998), the OSS paradigm can be seen as a provocative, yet fascinating, approach driven by ideological conviction and altruistic interests (Sharma ET AL., 2002). The third approach, CSD, is adopted by companies trying to infuse features from the community culture into their every-day practices of software development (Sharma ET AL., 2002). In representing formal and informal organizational structures, and in being approaches that to varying degrees involve the user community in the development process, the approaches offer different opportunities for the range of companies entering the distributed environment of software development.

6 Conclusions

In this paper, three approaches to distributed software development are identified and explored — global software development (GSD), open source software development (OSS) and community-based software development (CSD). Based on a comparison of three empirical cases, it is argued that the approaches embrace different characteristics in terms of: (1) nature of development approach, (2) communication structure, and, (3) coordination mechanisms. In identifying and exploring the three approaches, this research aims at helping both researchers and software developers in:

• Recognizing the different approaches to distributed software development that can be taken.
• Recognizing the differences between the approaches, hence, creating an understanding for the types of development situations to which they can be applied.

References

Bergqvist, M. and Ljungberg, J. (2001). The power of gifts: organizing social relationships in open source communities. Informations Systems Journal, 11, PP. 305-320.

Beyer, H. and Holtzblatt, K. (1998). *Contextual Design: Defining Customer-Centered Systems.* Academic Press: London.

Blanchard, A. L., and Markus, M. L. (2002). Sense of Virtual Community — Maintaining the Experience of Belonging. In *Proceedings of the 35th Hawaii International Conference of System Sciences* (HICSS), January 7-10, Big Island, Hawaii.

Carmel, E., and Agarwal, R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development. IEEE *Software*, March/April.

Dubé, L. and Paré, G. (2001). Global Virtual Teams. *Communications of the* ACM, 44(12).

Feller, J., Fitzgerald, B., and Van der Hoek, A. (2001). Making sense of the bazaar. In Proceedings of the First Workshop on Open Source Software. Feller, J., Fitzgerald, B., and Van der Hoek, A. (EDS.). *23rd* ICSE *Conference*, Toronto.

Feller, J. and Fitzgerald, B. (2002). *Understanding Open Source Software Development.* Addison-Wesley: London.

Gallivan, M.J. (2001). Striking a balance between trust and control in a virtual organization: a content analysis of open source case studies. *Information Systems Journal*, 11, PP.277-304.

Henfridsson, O.and Holmström, H. (2003). Developing E-commerce in Internetworked Organizations — customer involvement throughout the value chain in the case of the online computer game Clusterball. DATA BASE — Special Issue on *Developing E-Commerce Systems*, Current Practices and State-of-the-Art, VOL. 33, NR. 4.

Holmström, H. (2001). Virtual Communities as Platforms for Product Development — an interpretive case study of Customer Involvement in Online Game Development. In *Proceedings of 22nd International Conference on Information Systems* (ICIS), December 16-19, New Orleans, LA, USA.

Karolak, D.W. (1998). *Global Software Development — Managing Virtual Teams and Environments*. IEEE Computer Society Press: Washington.

Klein, H. K. and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS *Quarterly* 23(1), PP. 67-93.

Ljungberg, J. (2000). Open Source Movements as a Model for Organizing. *European Journal of Information Systems*, VOL. 9, NR. 3, PP. 208-216.

McDonough, E., F., Kahn, K., B., and Barczak, G. (2001). An investigation of the use of global, virtual and collocated new product development teams. *Journal of Product Innovation Management*, VOL.18, PP. 110-120.

Moon, J.Y and Sproull, L. (2000). Essence of distributed work: the case of the Linux Kernel. *First Monday*. HTTP://WWW.FIRSTMONDAY.ORG/ISSUE5-11/MOON/INDEX.HTML.

Namioka, A. and Shuler, D. (1993). *Participatory Design; Principles and Practices.* Hillsdale: New Jersey.

Nyberg, A-K. and Henfridsson, O. (2001). Going for the Online Customer — An Interpretive Case Study of Internetworked Customer Reach in Online Entertainment. In *Proceedings of the 9th European Conference on Information Systems*, June 27-29, Bled, Slovenia.

Orlikowski, W. (2002). Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organizational Science*, VOL. 13, NR. 3, PP. 249-273.

Raymond, E. S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly: Cambridge.

Scacchi, W. (2002). Understanding Requirements for Developing Open Source Software Systems. IEEE *Proceedings — Software Engineering*, 149(1), PP. 24-39.

Shankland, S. (1998). Linux shipments up 212 percent. CNET *News.com*, December 16. Accessed June 28 at HTTP://NEWS.CNET.COM/CATEGORY/0-1003-200-336510.HTML.

Sharma, S., Sugumaran, V. and Rajagopalan, B. (2001). A framework for creating hybrid-open source software communities. *Information Systems Journal*, 12, PP.7-25.

Sheremata, W.A. (2002). Finding and solving problems in software new product development. The *Journal of Product Innovation Management*, 19, PP.144-158.

Smith, P., G., and Blanck, E., L. (2002) From Experience: leading dispersed teams. The *Journal of Product Innovation Management*, 19, PP. 294-304.

Walsham, G. (1995) Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, (4), PP. 74-81.

Virtual community use for packaged software maintenance

Helena Holmström
Viktoria Institute • Gothenburg • Sweden

Brian Fitzgerald
University of Limerick • Limerick • Ireland

Abstract
This paper investigates the use of virtual communities for involving distributed customers in the maintenance of packaged software. On the basis of an empirical study it is suggested that virtual communities can be usefully leveraged for corrective, adaptive and perfective software maintenance. Specifically, the virtual community allowed for quick discovery of bugs and a rich interaction between developers and customers in the categories of corrective and adaptive software maintenance. However, while contributing also to the perfective category of software maintenance, this was the category in which several customer suggestions for modification were actually ignored by the developers. This implies that community use is indeed beneficial for maintenance related to coding and design errors as well as for maintenance of an adaptive character. However, it has limitations when associated with major changes such as software functionality addition or modification as those experienced in the category of perfective maintenance.

Keywords: Packaged software, software maintenance, virtual communities

1 Introduction

Despite research which suggests that software maintenance consumes between 40 to 75 percent of the total resources in software development (see E.G., Alkhatib 1992; Boehm 1981; Lientz and Swanson 1980; McKee 1984), maintenance appears not to be highly regarded by software programmers or their managers (Babcock 1987). One of the pioneers in the software field, Ed Yourdon, captured this well in the contention that for many programmers, maintenance was a fate worse than death, a view reinforced by Schneidewind (1987) who suggested that to be identified as working as a maintenance programmer was equivalent to being perceived as having bad breath. This results in the paradoxical situation that although maintenance of existing software is arguably more intellectually challenging than development of new software, the most junior and inexperienced programmers are frequently charged with the task. Thus, any initiative which can facilitate the maintenance task deserves to be examined closely.

Much research has been conducted on the maintenance topic, and a tripartite typology of corrective (repairing faults after delivery), adaptive (adapting the software to new operating environments) and perfective (adding to, or modifying, software functionality) maintenance is widely adopted (E.G., Alkhatib 1992; Lientz and Swanson 1980; McClure 1981). This research has identified a number of factors that could contribute to easing the maintenance task, including assessment of software maintainability (Vessey and Weber 1983), factors associated with software repair (Lientz and Swanson 1980), and various maintenance tools (Smith 1999). However, this research has not typically focused on possible expansion of the role of the customer, a notable exception being the study of Hirt and Swanson (2001) which investigated the expanded role of customers in the maintenance of ERP systems. Likewise, in the open source software (OSS) area, Schmidt and Porter (2001) investigated the role that users could play in debugging, documentation, mentoring and technical support. Given the suggestion that 60 percent of the time spent on a program modification request is consumed in locating the lines to change (Smith 1999), any help that the customer community can provide in elaborating the nature of the problem, and thus helping to identify the section of the program to be changed, could be very beneficial. While the difficulties of software maintenance have been the subject of much research to date, an important development in more recent times has been that maintenance increasingly takes place in the context of packaged software development. In this mode, customers form a diverse group who are often far removed from developers. Thus, all the traditional difficulties manifest in the maintenance process are further exacerbated. For example, a critical problem in software maintenance is the elicitation of changing customer needs and requirements (Nelson and Cooprider 2001). This is particularly true in relation to packaged software which is sold to pan-globally located customers. Recent research on innovation and distributed development suggests that innovation is stimulated by the diversity which can naturally arise by leveraging the expertise and diversity of geographically distributed customer groups (Chesbrough

2003). This is particularly important in the case of perfective maintenance, as the identification of new functionality requires both innovation and creativity.

Recognizing these problems, the primary research objective in this paper was to explore how virtual communities, as platforms for interaction, could be used to elicit changing customer needs and requirements in the maintenance process of packaged software and hence, involving distributed customers in the software maintenance process. While there is considerable research on virtual communities as beneficial to software development (see E.G. literature on OSS development by Raymond 1999; Feller and Fitzgerald 2000), this paper explores the specific deployment of virtual communities in the process of packaged software maintenance. Based on an empirical study of a computer game community, it is illustrated how virtual communities can support the software maintenance process for corrective, adaptive and perfective maintenance.

2 Background

2.1 SOFTWARE MAINTENANCE

Pressman (1997) has stressed the importance of distinguishing between the maintenance process and the software configuration management process. According to Pressman, software maintenance is a set of software engineering activities that occur after software has been delivered to the customer. Changes are made in response to changed requirements but the fundamental structure of the software remains stable. Software configuration management, on the other hand, refers to the set of tracking and control activities that are initiated when a software project begins and terminate only when the software is taken out of operation. Hence, software maintenance can be regarded as a subset of the software configuration management process, and in this paper we will focus on the corrective, adaptive and perfective maintenance processes.

In the corrective maintenance phase, coding errors, design errors and requirements errors are handled. While coding errors are relatively cheap to correct, design errors are more expensive as they may involve the rewriting of several program components. Most expensive however, are requirements errors since they might require extensive system redesign (Sommerville 2001). In contrast to a common belief, repairing system faults is not the most expensive maintenance activity. Studies have shown that only 17 percent of maintenance is concerned with correcting faults (Lientz and Swanson 1980). Rather, evolving the system to cope with new environments and to new or changed customer requirements consumes most maintenance effort.

Adaptive maintenance is required when there is a need to adapt the software to a different operating environment, for example if some aspect of the system's environment such as the hardware, the platform operating system or other support software changes, or if other environmental changes require the adaptation of the software. In a study by Lientz and

Swanson (1980), it was discovered that about 18 percent of the maintenance work was concerned with software adaptation.

Finally, perfective maintenance is concerned with software functionality addition or modification. This type of maintenance is necessary in response to changes in customer requirements. According to Lientz and Swanson (1980), 65 percent of the maintenance effort is distributed on functionality addition or modification due to changes in customer requirements. Hence, of critical concern to this process is the elicitation of changing customer needs and requirements. However, to elicit these is a complex process. Goals such as identifying system boundaries, identifying stakeholders and identifying different customer groups are important but inherently difficult to accomplish. In addition to this, it is often the case that customers find it difficult to articulate their needs and requirements in an early stage of the process (Nuseibeh and Easterbrook 2000). To help in this process, there are several elicitation techniques. Besides observations, questionnaires, interviews and analysis of existing documentation, there are group elicitation techniques, prototyping, model-driven techniques, cognitive techniques and contextual techniques (Byrd ET AL. 1992). Also, different techniques have been categorized as either informal or formal, where the informal approaches consist of face-to-face conversations between customers and developers while the formal approaches consist of structured documents that are produced and signed off by the customers (Sommerville 2001). Often, changes to the software are implemented iteratively and customers can be directly involved in testing new versions of the software. Requirements analysis which is conducted on the basis of actual experience with a real system artifact is a much richer experience and is more likely to yield more insightful and accurate requirements than the conventional model which typically requires customers to express their notional requirements in the absence of any detailed interaction with a real system artifact.

However, despite these techniques, the process of requirement elicitation and the ability to adjust to changing customer needs is still difficult. Furthermore, there has been a recent shift in software development processes and software products. To a large extent, software development is now performed by software vendors, and contrary to custom IS where made-to-order systems are built for specific customers, many software products of today are sold as packaged software, I.E. tradable products intended for mass use (Sawyer 2000). As recognized by Sawyer (2000), this will alter way we think about software development and certainly, this will have implications also for the process of software maintenance. While the same categories of maintenance still have to be accomplished, this has to be achieved in cooperation with a distributed customer group that never interacts physically with the software developers. This implies that the process of software maintenance is different — and perhaps even more complex — to that described in traditional software engineering literature. Given the considerable effort and cost of software maintenance, it is worthwhile exploring alternative approaches for interacting also with distributed customers. In this paper, we discuss how such interaction was achieved by using a virtual software community.

2.2 SOFTWARE COMMUNITIES

With interaction media such as email, chat and conferencing systems, computer networks of today allow for people to create a wide range of new social spaces. On the Internet, people engage in topic-specific discussions groups, play games, entertain one another and even work on complex collective projects (Smith and Kollock 1999) such as software development (Butler 2001).

In this paper, we focus on software communities, I.E. communities in which people interested in particular software products meet to collectively discuss these products and, in some communities, also participate in developing these. Primarily, these communities can be found in relation to software such as web infrastructure applications and computer games and collective action is related to the performance of different development tasks such as debugging, modification and improvement of that particular software. For example, in open source software (OSS) communities, world-wide communities of software developers engage in developing software that can be freely shared for review, reuse and modification. According to Sharma ET AL. (2002), the OSS model is a fundamentally new way to develop software and one that provides unique opportunities in terms of developer base and user input. In the Apache HTTP Project there are developers from Canada, Germany, Italy, the US and the UK (Fielding 1999), and recent studies on the Linux kernel development community show activity in more than 28 countries (Hermann ET AL. 2000). Based on the Linux case, OSS proponents argue that the model makes possible for quality software to be produced in a short period of time, with little cost, and by some of the best programmers in the profession (Sharma ET AL. 2002). This has also encouraged for-profit organizations to try to build business models around the open source paradigm and now companies such as HP, Intel, IBM ETC. are helping create an Open Source Development laboratory to promote OSS collaboration and growth. Typically, OSS communities develop Internet and web infrastructure applications such as for example the Apache web server and the Mozilla Web Browser. The development process is iterative (Sharma ET AL. 2002) and characterized by parallel development, prompt feedback to user and developer contributions, and the use of extremely rapid release schedules (Feller and Fitzgerald 2000). Furthermore, OSS community members value altruism, reciprocity and gift giving, and while financial reward is the main motivation in conventional software development this does not seem to be that significant for OSS community members. Instead, the personal benefit of using an improved software product is the driving force in OSS communities (Sharma ET AL. 2002).

Also, software communities are found in relation to computer game development. As recognized by Scaachi (2002), the release of Doom onto the Web in open source form in the middle of the 1990's began what is recognized as the landmark event that started the development and redistribution of open software game variants, so called PC « mods ». Mods are game variants that are created by small numbers of users who want to modify games instead of using them as they are provided. Today, the scope of mods has expanded to include entire new game types, game character models and skins (surface textures), levels (game play arenas) and AI game bots (in-game opponents). As in OSS communities, game community members value trust and reputation and to be generous with one's time, expertise and source code are valued traits of community participants (Pavlicek 2000).

In looking at these two examples, there is little doubt that software communities offer interesting opportunities in terms of involving customers in the software development process. While open source communities allow for users to access the source code and modify the software, other software communities allow for electronic discussion forums in which software users provide each other, and the software developers, with important feedback on the particular software. In such communities, users do not modify the software themselves but contribute to the development process in terms of knowledge they acquired when using the software. As recognized within the field of packaged software development, customer involvement is not common (Sawyer 2000), and when present, often in the form of intermediaries or customer surrogates (Keil and Carmel 1995). Hence, software communities can be seen as an interesting approach for involving distributed customers in the development and maintenance of software. Also, the examples above indicate an expansion of the traditional role of customers. In OSS development, for example, software users are often also the software developers, and while there are indeed hierarchies within open source communities, there is a high user dependency and hence, high user impact. In this paper, we focus on the maintenance process of packaged software. While there has been considerable research on software maintenance and how this task can be facilitated, this research has not typically focused on possible expansion of the role of the customer. Therefore, we take a closer look at the opportunity to use software communities to elicit changing customer needs and requirements and hence, expand the role of customers to more active participants in the software maintenance process.

3 Empirical setting and research process

3.1 DAYDREAM SOFTWARE

Daydream Software is a Swedish computer game developer with its foundations in Sombrero AB, a company focusing on software systems and hardware sales. Daydream Software was founded in 1994, and is currently focused on producing interactive entertainment. During the period of this study, January 2000 — October 2002, the company consisted of employees ranging from managers, administrative personnel and marketing people to game developers, graphical designers and web designers. With successful products such as Safecracker and Traitors Gate, Daydream has a large international customer base and well established customer communities around its products. In developing Safecracker and Traitors Gate, all software was developed in-house and then sold as packaged software in which distributors and publishers were important actors. As common in packaged software development (Keil and Carmel 1995), customer polls and market research reports helped the developers in getting information about customers' needs and requirements. Also, beta testing was performed by parts of the customer group in order to facilitate the development process and bring a complete product to the market. However, when released, both

Safecracker and Traitors Gate were static in the sense that customers could no longer influence the products. This was recognized by one of the developers at Daydream:

> *"Both Safecracker and Traitors Gate were static products without vivid customer communities. During development we got user feedback in terms of beta testing, but customers were not directly involved in any further modification of the games."*

Following on the success of these products, Daydream introduced Clusterball, a multiplayer computer game in which players fly around ships trying to collect balls and steal them from other players. In contrast to Safecracker and Traitors Gate, which were both commissioned work, Clusterball was the result of an in-house vision — the idea of an online sport accessible also via the Internet. An important point of departure for Daydream, and also for the inspiration for this study, was the explicit intention from the outset to utilize customer knowledge in the maintenance and improvement of the game. To this end, a virtual community was established which would cater for customer-developer and customer—customer interaction, not only during beta testing but also during the maintenance process. For Daydream, the community would allow for the continuous elicitation of customer needs and requirements, something that had been difficult to achieve when developing the two previous products. Also, the community would provide a mechanism for the customers to influence the maintenance process in terms of fault repair, software adaptation and software modification. This is a well-known, yet difficult challenge in traditional software maintenance, and the attempt by Daydream to use a virtual community represents a quite novel way to address it.

3.2 THE CLUSTERBALL COMMUNITY

Here we use Hamman's (2001) four criteria to characterize virtual communities, namely group of people, shared social interaction, common ties and shared social area. These are used to describe the characteristics of the Clusterball community.

A group of people

The Clusterball community is a game community consisting of members from northern Europe and the us in the main. Depending upon previous game scores, each member is categorized according to the official Clusterball ranking, ranging from « newbie », « bellboy » and « trainee » to « master », « grand master » and « cluster king ». In total, there are 20 different ranking categories and members with the highest rankings are well-known and celebrated members in the community. Together, they engage in discussions concerning Clusterball, and on a regular basis they arrange tournaments and team-play as well as tutorials and training sessions for all Clusterball beginners.

In order to stimulate the interaction between customers and developers a « community manager » was appointed in August 2000. This person was responsible for responding to — and implementing — suggestions put forward by the customer community. This helped to ensure that the community was nurtured, and that valuable feedback was not lost.

Not surprisingly, many of the developers at Daydream are active community members. This is a feature of many open source software projects also, in that invariably the software developers are themselves also actual users of the software, something that facilitates in the process of building a community (Feller and Fitzgerald 2000).

Shared social interaction
With approximately 17,000 postings distributed among two different forum tracks over a three-year period, the Clusterball community provides an active discussion forum for the development and the modification of the game. As recognized by Baym (1998), the communicative style of participants in such communities are often oriented around common interests and practices even before they enter the computer-mediated world, and often the members adhere to certain norms of rational discourse. In this case, the technology becomes an enabler of already established physical communities — a description that is very apt for the Clusterball community.

However, CLUSTERBALL.COM is not the only place where the Clusterball community meets. Besides this forum, there are fan websites (websites developed by community members themselves) that offer forums and chat rooms for community members, and team websites where different teams meet and sign up for tournaments. One of the most impressive fan websites is BALLSNATCHERS.COM which was originally developed exclusively for Clusterball by two of the players, and which is now maintained and further developed by a team of Clusterball players from all over the world. Here, players have their own « hall of fame » (player/team victory announcements), a « haiku corner » (player poems) and a « player gallery » (player portraits).

Common ties
The common interest in the Clusterball community is computer games in general, and Clusterball in particular. In the different forums, community members discuss configuration and installation problems as well as tournaments, team-play and how to improve the game. At CLUSTERBALL.COM there is the « technical » and the « general » forum, and at BALLSNATCHERS.COM there is a specific forum for beginners called « young wings » where new players can post any questions they might have to the rest of the community. Also, there is a « chat-and-gossip » forum in which players discuss anything that comes to mind.

The devotion and motivation among community members can also be seen in the activities they organize. For example, there are several Clusterball Schools for beginners (see for example Ootpek's Clusterschool, Kronix Tips and Lava-Lava's Clusterball Tips at WWW.CLUSTERBALL.COM), a Skin Tutorial in relation to a skin site on which players upload their individually designed skins so that other players can download and use them, and a testimonial site where Clusterball players share experiences from their initial contact with Clusterball.

Shared area
To communicate, the Clusterball community members send postings to electronic fora consisting of several different tracks. In these, headings are shown for all topics, and all

postings are presented as threaded lists. Also, there is a chat so that people can meet before the game, as well as after, to discuss issues concerning that particular gaming session. In addition to this there are the fan websites where several other fora and chats can be found and where many of the Clusterball players spend time on a regular basis.

3.3 RESEARCH METHODOLOGY

Research design
The research outlined in this paper is part of a longitudinal interpretative case study (Walsham 1995) conducted at Daydream Software between January 2000 and May 2001. This study consisted of an exploratory study, in which we sought an initial understanding of the company, an in-depth study involving participant observation at the research site and a complementary data collection phase in which qualitative interviews and a web survey were carried out. In addition to this, a follow-up study was conducted between June and October 2002. In this, additional interviews were held and we sought to deepen our understanding of the particular context of Daydream Software and the way in which a virtual community was used for improving customer-developer interaction in the software maintenance process.

Data sources and applicability of results
In this particular paper, our objective was to investigate to what extent virtual communities can be used for involving distributed customers in the maintenance process of packaged software and hence, how they might address the problematic issues as identified earlier in the paper. To do this, four categories of different empirical data were extracted from the Daydream study (TABLE 1):

Empirical data	*Description of empirical data*
Postings	Written messages revealing customer needs and requirements as well as suggestions for software improvements.
Patch specifications	Technical specifications including modifications and new features that were implemented in each of the new software versions that were released as responses to customer needs and requirements.
Web survey	A web based questionnaire including questions on community use and to what extent customers felt that they could influence the software maintenance process.
Interviews	Qualitative interviews revealing the developers' apprehension of the development process of Clusterball and to what extent the community allowed for customers to participate in this process.

TABLE 1. The different categories of empirical data that were extracted from the overall Daydream study for the purpose of this paper

Firstly, postings to the technical forum at CLUSTERBALL.COM were analyzed with regard to discussion theme and result in terms of modifications to the software. The postings included those sent to the forum between the release date July 17, 2000 and May 2001, when the complementary data collection phase was finished. During this period, 1,116 messages were posted to the technical forum, of which the major part were sent between July and December 2000 when the game was still new and when there were a lot of technical issues to handle. In reading the postings, special concern was taken to those reflecting customer needs and requirements and whether these were implemented in the coming patches.

Secondly, patch specifications were studied to learn about the changes that were implemented in new versions of the software and whether these could be related to the postings in the forum. To do this, specifications of six different patches were analyzed. The patches included in this study were released on July 18, August 25, October 19 and December 20, 2000, and February 22 and April 29, 2001. While parts of these specifications were found at WWW.CLUSTERBALL.COM, other parts were obtained directly from the developers at Daydream.

Thirdly, a web based survey was sent out to 200 Clusterball community members, ranging from « Newbies » (not very experienced players) to « Ring Kings » (very experienced players). The survey was sent out as part of the complementary data collection phase in October 2000 and consisted of questions regarding the use of the community and the way in which community members felt that they could influence the maintenance work of Clusterball. With a response rate of 52 percent the survey helped us in exploring community use and community influence in the software maintenance process.

As a complement, qualitative interviews were conducted with the lead programmer and one of the graphical designers. These interviews were conducted during the follow-up study, and in these, the interviewees were asked to look back on the maintenance process of Clusterball and evaluate how, and in what situations, customers in the virtual community contributed to the different categories of maintenance. Each interview lasted for about 1.5 hours and they were both recorded and transcribed.

In terms of generalizability, case study research is often criticized for being non-representative (Walsham 1995), and therefore of limited use outside its specific context. However, from an interpretive position, representativeness in a statistical sense is not a key goal, but instead the plausibility and cogency of the reasoning used in describing the results from the case and in drawing conclusions from these results (Walsham 1995). In our study, we explore the use of virtual communities for involving distributed customers in the maintenance process of packaged software. In this, our goal is not to present generally applicable results. Undoubtedly, not all software communities are like the computer game community presented here, and not all software has characteristics similar to those in a computer game. The extraordinary motivation level of Clusterball community members and hence, the benefit of involving them as active participants in the maintenance process of Clusterball may not be applicable in other communities or in relation to other software products. Still, the Clusterball case constitutes an interesting example that illustrates the potential use of virtual communities for software maintenance and the extended customer role that is associated with this. While it might be difficult to translate all its aspects to the maintenance process of other software products, we provide a detailed account involving specific implications in this particular domain of action (Walsham 1995). In doing this, the study adds to our understanding of virtual community use and for what particular categories of software maintenance such an approach might prove useful.

4 Overview of Clusterball patches

The Clusterball game uses the 3DGM graphical engine. It is programmed in C++, modeled in Java and the Sourcesafe product was used for version control for the project. To implement changes in customer needs and requirements, Daydream released six patches to the game. As early as July 18, 2000, only one day after the official release, the first patch to Clusterball could be downloaded from the Internet. In addition, the second patch was released on August 25. Together, these patches solved many of the initial installation problems and start-up errors as well as host errors that were recognized by customers.

On October 19, 2000, the third patch was released. This patch included several adjustments and modifications as suggested by the customers. For example, the patch included:

• Replay and recording
• DNF feature (players could still get points even if they « did not finish » the game)
• Capability to lock a server on a min/max ranking basis (to avoid for newcomers to play against too highly skilled players or for highly skilled players to play against too novice players.
• Pre-game chat (chat to other players while waiting to join a game)
• Ranking for team-play
• Longer chat lines in the in-game chat

On December 20, 2000, the fourth patch was released. This was an A2D driver patch that was released to solve a problem related to customers using the ATI RAGE PRO LT graphics card. The A2D driver patch installed all the necessary A2D drivers for the graphics card and thus provided support for customers using this particular graphics card.

In addition to this, the fifth patch was released on February 22, 2001. This was the GL SETUP patch which detected what kind of graphic card the user had and then downloaded and installed the latest drivers for that particular card.

Finally, the sixth patch was made available on April 29, 2002. This patch included bug fixes, software adaptations and functionality additions. Among other things, these new features were included:

• Improved support for joysticks including « twist handle » functions
• LAN-play without restriction of Internet access for host
• Improved artificial intelligence (AI) in the training (offline) mode

Also, the following bugs were addressed:

- Crash bug in the pre-game chat
- Freeze bug when viewing replays
- Throttle bug on joysticks
- DNF bug in match history
- Sound volume bug
- Font problem in chat

In studying the content of the Clusterball patches it is evident that many of the improvements that were made to the software originated in customer suggestions as reflected in the community postings. Below, we discuss this process in more detail, illustrating the community contribution to the various categories of software maintenance.

5 Community contribution to Clusterball maintenance

Clusterball was released on July 17, 2000 and made accessible to customers all over the world. At this point, improvement, in terms of software maintenance began. In the following discussion, community postings are analyzed in order to illustrate the use of the virtual community in the maintenance process of Clusterball.

5.1 CORRECTIVE MAINTENANCE

Corrective maintenance is concerned with software fault repair, coding errors, design errors and requirements errors. As recognized by Sommerville (2001) these types of errors are not the most expensive to correct. However, they need to be attended to on a continuous basis. In this process, Daydream got significant help from the community. Consider these community postings — all regarding different error messages:

> *"A few times now I've had the game just hang. It's always right after a game when it says « time limit reached » or on the loading screen before a game starts. It will just stay at those screens forever and nothing will happen. I was curious if this was related to* WIN2K *or something else. Machine is* P2-450, 348 RAM, VOODOO3 3000, WIN2K PRO. *I've installed the host error patch as well though this happens when I'm joining a game not hosting."*

> *"Right when the loading screen appears I get an illegal operation and I have to close it. This happens every single time. I have a* DIAMOND VIPER V550, *running* 1280 X 1024 32BIT, *and* WIN 98. *I have had some problems with other games not switching to* DIRECT 3D *mode but nothing like this. Please help, I'm very annoyed."*

> *"Hi, My crashes end with:* CLUSTERBALL *caused an invalid page fault in module* CLUSTERBALL.EXE *at* 015F:0054E7E4. *It crashed mid-game. I have a* 450 ATHLON, 256 MB, GEEFORCE 256. *Any suggestions? Thanks."*

These errors were handled in the first and second patches. Most installation and start-up problems were solved in the first patch, and in the second patch, released on August 25, host errors were solved.

Additional software faults that were recognized by customers concerned font problems, sound volume problems and a crash bug that appeared when too many customers joined the pre-game chat:

> *"I like the new patch, but damn, I can't read anything when joining the chat. I have a 19 inch monitor but I still can't read that crappy font. Also the sound is still a problem in* XP *with* SB LIVE *sound card".*

This posting got a quick response from one of the developers at Daydream:

> *"... the soundcard thing is out of our reach, let's just hope that Creative will update their crap drivers for* XP *soon. I've got this sound volume problem with lots of other games in* XP *as well...see what I can do."*

Regarding the crash bug in the pre-game chat, this was mentioned by one of the customers:

> *"When there are too many « activities » going on in the pre-game chat room, Clusterball has a tendency to crash. I didn't note the error message though."*

The font problem, the sound volume problem and the crash bug in the pre-game chat were solved in the sixth patch that was released in April 2002. In this patch, fixes for these bugs were included, together with several additional features as requested by the customer community.

5.2 ADAPTIVE MAINTENANCE

Adaptive software maintenance is required to adapt the software to different operating environments, for example if some aspect of the system's environment such as the hardware, the platform operating system or other support software changes, or if other environmental changes require the adaptation of the software (Sommerville 2001).Consider the following community postings, all concerning software adaptation:

> *"I wish the standard « control setting » was better. At present, getting a good control setting is too much a case of trial and error. But if people have problems with this they don't play."*
>
> *"It would be better if there was support for more video cards..."*
>
> *"I would like to see the Mac version of the game."*
>
> *"It would be helpful, if it was possible, to run Clusterball in Windows Mode, or at least if you could minimize it."*

> *"I think you should integrate an* IRC *client into the software so that you could access the Clusterball channel from inside the client."*

A common theme across these postings was the desire for adaptation of the software to other operating environments or to be able to play the game using other configurations.

Another problem recognized by customers in the community was the joystick problem:

> *"I tried playing again today after a week... and sometimes the stick works okay. But the last few games I was only trying to stay on course (slamming the platforms and bumping into the equipment houses. This takes so much time, and other pilots start taking over my route, and then when I fly normally again, I arrive late everywhere. It occurs suddenly, and then it stops. Because of these problems other players have a lot of chances and I can't avoid them shooting because I can't fly properly. Does anyone have these problems with their joystick or am I the only one?????? Can someone help me out here????"*

Furthermore, throttle problems with the joystick were discovered:

> *"I've just bought the* SAITEK CYBORG 3D *joystick which has solved the jerky controls of my previous low budget version. The problem is the throttle doesn't seem to give full speed and the response to change direction* ETC. *is very slow. I use standard* PRO *settings. The joystick seems to be calibrated and profiled correctly but I can't get round these problems. Any ideas anyone?"*

The joystick problems were all solved in the sixth patch which also handled the DNF (did not finish) issue that was flagged by one of the customers:

> *"My experience of* DNF *is that I send data, but don't receive anything, and when enough time has passed, my computer evaluates this as the server having gone down (not closed, that is a different message!), and ends the session. Is there a way for us* DNF*-targeted to ignore « Bad connection » for a longer period, maybe set this in the configuration file?"*

On December 20, 2000, the fourth patch was released. This was an A2D driver patch that was released in relation to graphic card problems that resulted in strange coloring of the balls in the game. This problem was identified by one of the customers:

> *"I got an* ATI RAGE 128 *graphics card. I'm not sure what you mean about the « environment map ». But when I play Clusterball, all the ships and balls are black. I need some help. I can still play but it is very annoying. Well thanks for the help."*

In responding to this, one of the developers elaborated further on the nature of the problem:

> *"I have actually seen this « black ball » phenomenon happen during development. I think it was with very old drivers on a* RIVA TNT *card. Since you want* FULL *support I'll just start asking my (huge) line of questions:*
> *1. Have you installed the latest* OPEN GL *drivers for your* ATI RAGE 128*?*

> *2. Could you check in the Control Panel-> Display->Adapter that* OPEN GL *is chosen the* DEFAULT *renderer (important!)*
> *2.2 Also, while in the Display settings, what level of* 3D *acceleration is set,* FULL, *75%, 50%, 25%, or what?*
> *3. There is a possibility that the game tries to run software renderer instead of* OPEN GL*… is your graphics « grainy » like software rendering?*
> *4. Do you have* DIRECT X *7.0 installed?*
> *5. What is your computer name, processor speed,* ETC.*?*
> *6. If you open the* CONFIG.CFG *file in Wordpad, what value does it say after renderer?"*

Other community members, using other graphics cards, were also involved in the discussion:

> *"Dear Clusterball Support: Is* CD *supposed to work with the* VOODOO5 FSAA*? Whenever I have it enabled, starting a match freezes my system 80% of the time online, and about 20% when offline training.* FSAA *works 100% fine on* ALL *of my other games, online and offline."*

> *"Clusterball is not working so well with the new driver from* NVIDIA*. I'm using a* TNT ULTRA 2 *and have tested the new drivers. I installed it and I'm using the old one from January 2000."*

To solve the graphics cards problems, the A2D driver patch installed all the necessary A2D drivers and hence, provided support for users using an ATI RAGE PRO LT graphics card. In addition, the fifth patch was released on February 22, 2001. This was the GL SETUP patch which detected what kind of graphic card the customer had and consequently downloaded and installed all the latest drivers for that particular card.

5.3 PERFECTIVE MAINTENANCE

Perfective maintenance requires functionality addition or modification in response to changes in customer needs and requirements. In the Clusterball community, changes in customer needs and requirements were reflected in postings concerning for example, the ranking system, the need for a comprehensive chat feature and the desire for a « player search » function. The following postings all exemplify customers' suggestions for improving the ranking system:

> *"I would like to see tournaments for middle class rankings. There are tournaments for new people and for high class players, but the middle men are left out."*

> *"I would like to improve the match making — to allow the possibility to find other players closer to my skill level."*

> *"I would like to be able to set a minimum and maximum player ranking when I host a game. In this way, a Newbie game will really be for Newbies, experts won't come along and thrash everyone. Similarly, a group of experts won't have to worry about a raw « what do I do with these balls? » beginner unbalancing team play."*

To some extent, these problems were catered for in the third patch in which the functionality to lock a server on a minimal/maximal ranking basis was implemented, and hence, unbalanced match-making could be avoided.

Also, the need for elaborate chat features was expressed:

> *"It would be great to have a chance to chat with the experts. The « Ring Kings » could participate and give the « Newbies » some hints live."*

> *"I would like to have the possibility to talk to other players while waiting for a game."*

> *"There needs to be a better chat function in the game. The one that is there in this version is really bad — nobody sees it."*

In response to these postings, a pre-game chat was implemented in patch number three. Using this, players could talk to each other while waiting to join a game and they could exchange experiences from previous gaming sessions. Also, longer chat-lines in the in-game chat were implemented to improve the overall chat function. Finally, the need for a player search was flagged by the customers:

> *"A « player search » would be helpful. That way one could find a friend who is somewhere else in the ranking system."*

> *"What I miss is some sort of « player search » where you could find out more about a specific player, like for example* E*-mail, ranking, score, games played, where he/she lives and so on…"*

Contrary to most other suggestions, the player search was not implemented in any of the patches. The reason for this could not be found in any of the developers' postings to the community. Hence, postings regarding the player search can be seen as suggestions for future software improvement, something that was also the case for the following postings:

> *"Make more then just the ship playable, most other games have more than one model to choose from. It doesn't have to be that different, but still another model to choose from. Maybe there could be a model editor where players could make their own ships…"*

> *"Could there be a « viewer system » so that my friends could watch other people play before they participate themselves? The game would be more like a real sport if it was viewable on* TV *or the Internet."*

> *"Make the venues change weather sometimes. The sun isn't always shining. Perhaps a change in wind could make the venue Egypt more difficult."*

> *"I would like to see a password for the server when hosting a match. Setting the number of games or how long time the dedicated server should run. It would also be great if it was possible to send messages to the players when running a dedicated server."*

Interestingly, a similar phenomenon was reported in the Mockus ET AL. (2000) case study of the open source Apache web server. In their study, they found that 75 percent of suggestions for software modifications are ignored. This is quite common in open source projects, and it has been suggested that a meritocracy exists whereby the privileged few at the core control almost exclusively the ongoing development of the projects. Although the Clusterball case is not an open source project, there seem to be parallels with the phenomenon as reported on in the Mockus ET AL. (2000) study.

Despite the fact that suggestions, as those presented above, were never implemented during the period of this study, they can still be seen as important for Daydream in the maintenance work of Clusterball. Besides revealing suggestions for future software improvements, the postings reveal community engagement and community interest in the software that was produced.

6 Discussion

Based on the empirical findings in the Clusterball case, it is suggested that virtual communities, as platforms for interaction, are beneficial to the maintenance process of packaged software in all three of the maintenance categories.

First, in the corrective maintenance process, concrete descriptions on specific software faults were obtained from the users. In the postings, detailed error messages and full descriptions of hardware configurations were included to facilitate the bug tracing activities conducted by the developers. This process of software fault repair can be compared with the different automatic fault reporting systems that are included in many software products. In a similar fashion, customer suggestions were registered and attended to, and without any significant developer—customer interaction, the results could be found in additional software patches. However, while the community could be used as any ordinary bug reporting system, there was also the opportunity for developers to either give personally customized answers to each contributor, or to post responses to the community forum so that anybody interested could learn from the answer. From the customers' point of view, there was also the opportunity to have other customers commenting on the particular software fault and in what different situations it appeared. As can be seen in the Clusterball case, this allowed for an open discussion between customers and developers — and between customers — in which there was the possibility of finding not only the origin of a particular software fault, but also the different use circumstances in which this fault was evident. Furthermore, the discussion in section 5.1 reveals how triangulation occurred in that different customer reports helped to refine the location and cause of errors. As indicated in literature on requirement acquisition (Byrd ET AL. 1992), interactive dialogue is often required to establish the precise nature of a software fault, and the triangulation process described above helps to ensure such dialogue takes place. When one considers that estimates suggest that 60 percent of the time spent on a modification task is consumed in finding the location of the lines to be changed (Smith 1999), this detailed troubleshooting triangulation could be very beneficial indeed.

Secondly, in the process of adaptive maintenance, community postings revealed different user configurations, different hardware and software equipment that customers used and how the software could be adjusted to suit the different operating environments represented. In similar fashion as with postings regarding software fault repair, postings on software adaptation included hardware specifications and configuration details. In addition to this, however, software adaptation postings also included suggestions for future improvements of the game, and adjustments that would be necessary in order to meet the requirements presented by other software and hardware equipment. In this, community postings revealed not only information important for the maintenance of Clusterball, but also information about the dynamic relation between Clusterball and other software and hardware configurations. As recognized by Norvig and Cohn (1997), however, the word « maintenance » can be misleading when referring to adaptive changes such as those reported on here. According to these authors, the term maintenance can give the impression that the software has somehow degraded, and needs to be refurbished to its original condition. This is misleading since software programs, such as a computer game, do not degrade. They remain the same, while the environment and equipment around the program continuously changes. Thus, adaptive maintenance is really a process of upgrading or improving the software to meet the needs of the changing environment. In the Clusterball case, such improvement was evident in, for example, additional support for new joysticks and enhanced support for video cards.

Thirdly, in the process of perfective maintenance, innovative customer suggestions regarding the ranking system, the pre-game chat, and balanced match-making could be found. Taken together, these postings could be understood as recommendations for new system capabilities, either by adding new functionality or to modify existing functionality. However, while there was indeed a demand for customer suggestions, this was also the maintenance category in which suggestions were largely ignored. This is somewhat unfortunate as identifying new functionality is a major difficulty in conventional software maintenance (CF. Cusumano and Selby 1997), yet this is well catered for in this virtual community model, a feature of open source project also (Mockus ET AL. 2000).

While being recognized as important in the Clusterball case, suggestions regarding additional play models, a viewer system and changing weather in the venues were never implemented. This suggests that there were limitations in what the customers could influence. This is in accordance with one developer's view on the community and its importance in the maintenance process:

> *"I read the postings, but the main parts of the new features that are implemented are the result of our own ideas. We already know what we would like the game to be like."*

Accordingly, one of the customers commented:

> *"I don't think that I can influence the game itself, but more like little improvements and small features."*

While the above statements might suggest that there was no opportunity for customers to influence the maintenance process in terms of functionality addition or modification, it is important to recognize the context in which Clusterball was developed. Since Clusterball was not commissioned work but instead an in-house project, the developers had very strong ideas about the storyline and the overall design of the game. As indicated above, in some situations the developers' already knew how they would like the game to evolve and hence, there are reasons to believe that they didn't want their own storyline to be subject for too much external negotiation. Also, in developing software that is distributed to customers all over the world, there are additional actors such as distributors, publishers and vendors to consider in the development as well as in the maintenance process. As recognized by Zachary (1994), packaged software development is an activity driven by time-to-market demands and tight release schedules, making major changes or improvements difficult to attain depending on the surrounding circumstances in the software development environment. As illustrated in the Clusterball case, the process of software functionality addition or modification seems to be the process in which external actors, or strong in-house ideas, play an important role. As a result of this, many of the suggestions that were provided by customers were never implemented. However, while this implies that community use is limited in the perfective category of software maintenance, the community could still be used for the elicitation of innovative ideas important for future software improvement. Bergin and Keating (2003) have identified the need for an explicit model for software maintenance, and certainly, a model which would help the Clusterball developers harvest the wealth of useful suggestions from the virtual customer community would be very useful. This would also allow the strategic planning of which new features to implement, and some positive reinforcement could be provided to the customer community through, for example, the publication of a planned release schedule. In such a model, the community manager role within Daydream would also help ensure that such activities take place.

In looking back at the empirical material, there are certain benefits that can be associated with community use for software maintenance. A common feature in all maintenance categories is the community supportiveness, I.E. community members' willingness to help other community members in solving problems. As shown in the Clusterball case, customers mutually engage (1998) in helping each other and customer—customer interaction is a prominent feature of the community. In a community, getting help means giving help and the more people that get involved the better. This also reduces the workload of the developers somewhat since customers rather help each other before asking the responsible developer. In addition to this, the Clusterball case reveals community enrolment, I.E. community members' willingness to engage new members to the community. While being positive for the overall community atmosphere as well as for community activities such as tournaments, training-sessions and forum discussions, this also facilitates for the software firm in attracting new customers. In research within the field of packaged software, customer involvement has been recognized as important for improving the development process [Sawyer 2000; Keil and Carmel 1995). Indeed, many of the best suggestions for product improvements often come from customers (Von Hippel 1986). Recognizing this, the potential of having community members enroll new customers will benefit not only software development but also software maintenance in terms of an enlarged customer base and hence, increased customer feedback on software adjustments and improvements.

Certainly, the degree of supportiveness and enrolment may vary over time, and also between different communities, but the basic idea of having a group of people willing to support and involve each other is beneficial to the overall process of software maintenance. Part of the success of rapid application development (RAD) approaches has been attributed to the fact that some of the development task, for example documentation and testing, is devolved to the general customer community (Fitzgerald 1997). Likewise, it is often the case that developer—customer relations become frayed in traditional maintenance (Smith 1999), and the positive atmosphere between developers and customers generated in the Clusterball virtual community bodes well as a model.

7 Conclusions

This paper has examined the specific use of virtual communities for involving distributed customers in the maintenance process of packaged software. On the basis of an empirical study of a computer game community, it illustrates how virtual communities can be used in the process of corrective, adaptive and perfective maintenance.

In the corrective process of software fault repair, virtual communities facilitate bug tracing activities by allowing for the continuous elicitation of particular software errors in relation to individual customer configurations. Also, detailed troubleshooting help to refine the precise nature of the bug, and help to pinpoint its exact location. In the adaptive process, virtual communities bring forward customers' view on future versions and adjustments necessary to adapt the software to requirements put forward by other equipment. Finally, in the perfective process, virtual communities allow for the elicitation of innovative ideas for future improvements and new functionality. However, while we suggest that virtual communities prove useful for involving customers in all three categories of software maintenance, it seems that customer impact is most evident in relation to corrective and adaptive maintenance. Due to external requirements as proposed by actors such as distributors, publishers and vendors, or strong in-house ideas as proposed by the software developers themselves, customer suggestions have less direct impact in the category of perfective maintenance. In accordance with Bergin and Keating (2003) we suggest that an explicit model for software maintenance could allow for the suggestions for new functionality to be better recorded and for positive reinforcement to be provided to the virtual community.

Finally, the study suggests that there are certain benefits that can be associated with community use for software maintenance. First, community supportiveness, I.E. community members' willingness to help other community members in solving problems is identified as important for reducing the workload of the developers. Second, community enrolment, I.E. community members' willingness to engage new members to the community, is identified as central for attracting new customers to the community and hence, provides software developers with increased customer feedback in their everlasting process of software maintenance.

References

Alkhatib, G. (1992). The maintenance problem of application software. *Journal of Software Maintenance: Research and Practice*, VOL. 1, PP. 83-104.

Babcock, C. (1987). Staffers seek bolstered image. *Computerworld*, VOL. 21, NO. 20, P.8.

Baym, N. (1998). The Emergence of On-line Community. In Jones, S. (ED.) *CyberSociety 2.0: Revisiting computer-mediated communication and community*, PP. 35-68. Newbury Park, CA: Sage.

Bergin, S. and Keating, J. (2003). A case study on the adaptive maintenance of an Internet application. *Journal of Software Maintenance and Evolution: Research and Practice*, VOL. 15, NO. 4, July/August 2003, PP. 245-264.

Boehm, B. (1981). *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, New Jersey.

Butler, B.S. (2001). Membership size, communication activity and sustainability: a resource-based model of online social structures. *Information Systems Research*, VOL. 12, NO. 4, PP. 346-362.

Byrd, T.A., Cossick, K.L. and Zmud, R.W. (1992). A Synthesis of Research Requirements Analysis and Knowledge Acquisition Techniques. MIS *Quarterly*, March, PP. 117-138.

Chesbrough, H. (2003). *Open Innovation*. Harvard Business School Press: Cambridge, MA.

Cusumano, M and Selby, R. (1997) *Microsoft Secrets*. HarperCollins: London.

Feller, J. and Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. In *Proceedings of the 21th International Conference of Information Systems* (ICIS), Brisbane, Australia.

Fielding, R. T. (1999). Shared leadership in the Apache project. *Communications of the* ACM, VOL. 42, NR. 4.

Fitzgerald, B. (1997). A preliminary investigation of RAD in Practice. In Wood-Harper, A. T., Jayaratna, N., and Wood, J. (EDS.), *Methodologies for Developing and Managing Emerging Technology Bases Information Systems*, Springer-Verlag, UK, PP. 777-87.

Hamman, R.B. (2001). Computer Networks Linking Network Communities. In Werry, C., and Mowbray, M. (EDS.), *Online Communities*. Prentice Hall: London.

Hermann, S., Hertel, G. and Niedner, S. (2000). Linux study: first results. Linux study home page, HTTP://WWW.PSYCHOLOGIE.UNI-KIEL.DE/LINUX-STUDY/WRITEUP.HTML. (Accessed April 24, 2001).

Hirt, S and Swanson, E.B. (2001). Emergent maintenance of ERP: new roles and relationships. *Journal of Software Maintenance and Evolution: Research and Practice*, VOL. 13, PP.373-397.

Keil, M. and Carmel, E. (1995). Customer-Developer Links in Software Development. *Communications of the* ACM, VOL. 38, NO. 5, PP. 33-44.

Lientz, B. P. and Swanson, E. B. (1980). *Software Maintenance Management*. Addison-Wesley: Reading, MA.

McClure, C. (1981). *Managing Software Development and Maintenance*. Van Nostrand Reinhold Company: New York.

McKee, J. R. (1984). Maintenance as a function for design. In *Proceedings of* AFIPS *National Computer Conference*, Las Vegas.

Mockus, A., Fielding, R. and Herbsleb, J. (2000). A case study of open source software development: the Apache server. In Proceedings of 22nd *International Conference on Software Engineering*, PP. 263-272.

Nelson, K., M and Cooprider, J., G. (2001). The relationship of software system flexibility to software system and team performance. In *Proceedings of the 22nd International Conference on Information Systems* (ICIS), New Orleans, USA.

Norvig, P. and Cohn, D. (1997). Adaptive Software. PC AI *Magazine*, VOL. 11, NR. 1.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of International Conference on Software Engineering* (ICSE), 4-11 June 2000, Limerick, Ireland.

Pavlicek, R. (2000). *Embracing Insanity*. Sams Publishing: NY.

Pressman, R. (1997). *Software Engineering: a practitioner's approach*. European Edition. McGraw Hill: New York.

Raymond, E., S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly: Cambridge.

Sawyer, S. (2000). Packaged software: implicatopns of the differences from custom approaches to software development. *European Journal of Information Systems*, 9, PP. 47-58.

Scacchi, W. (2002). Understanding the Requirements for Developing Open Source Software Systems. IEEE—*Software*, 149(1), 24-39.

Schmidt, D. and Porter, A. (2001). Leveraging open source communities to improve the quality & performance of open source software, in Feller, J., Fitzgerald, B. and van der Hoek, A. (EDS.) Making Sense of the Bazaar: The 1st Workshop on Open Source Software Engineering, *23rd International Conference on Software Engineering*, Toronto, Canada, May 2001, PP.52-56.

Schneidewind, N (1987). The state of software maintenance. IEEE *Transactions on Software Engineering*, VOL.13, NO.3, PP.303-310.

Sharma, S., Sugumaran, V. and Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal*, 12, PP.7-25.

Smith, D. (1999). *Designing Maintainable Software*. Springer: New York.

Smith, M., A. and Kollock, P. (1999). *Communities in Cyberspace*. Routledge: New York.

Sommerville, I. (2001). *Software Engineering*. Addison-Wesley: Harlow, UK.

Vessey, I. and Weber, R. (1983). Some factors affecting program repair maintenance: an empirical study, *Communications of the* ACM, VOL.25 NO. 7, PP.128-134.

Von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science*, VOL. 32, NO. 7, PP. 791-805.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, VOL. 4, PP. 74-81.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning and Identity*. Cambridge University Press: Cambridge.

Zachary, G. (1994). *Showstopper: The Breakneck Race to Create Windows* NT *and the Next Generation at Microsoft*. The Free Press: New York.

Improving packaged software through online community knowledge

Helena Holmström
Viktoria Institute • Göteborg • Sweden

Ola Henfridsson
Viktoria Institute • Göteborg • Sweden

Abstract
Packaged software development (PSD) is largely a knowledge intensive activity. Thus, it depends on the organizational capability of developing and combining market and domain knowledge into timely and competitive software products. Given customers' situated knowledge of the software, software firms increasingly seek new ways to involve customers in their software development activities. As highlighted in the literature, one alternative path for doing this is to use online communities. However, there exists little empirical research that examines the role that communities can play in the commercial endeavor of developing packaged software. To address this omission, this paper examines the benefits and limits of online community use in PSD. The contribution of the paper is a model of community use in PSD. This model describes such use as three iterative and interrelated processes, unfolding at the intersection between commercial software firm practices and voluntary community participation.

1 Introduction

Packaged software is an increasingly important form of information technology. Already in 1998, packaged software was the fifth largest industry in the US (Sawyer 2000), and it is now widely used by both business organizations and consumers. Sold by vendors, distributors or stores, packaged software (also known as shrink-wrapped, commercial off-the-shelf or commercial software) can be distinguished as tradable software products that are designed to be easily installed and to interoperate with existing system components (Abts 2002).

Developing packaged software is largely a knowledge intensive activity (Clegg ET AL. 1996), driven by time-to-market demands in that breaking new ground is conceived critical for competitive advantage (Zachary 1994). Such development is conducted by developers that typically hold line positions making them central to firm performance (Sawyer 2000). Rather than holding supportive staff functions, packaged software developers are more center stage than most IT staff in that they often possess the core competence central to the software firm's competitiveness (CF. Hamel and Prahalad 1994).

Given the centrality of packaged software development (PSD), software firms continuously seek new ways to improve their development processes (Humphrey 1989; Mathiassen ET AL. 2002). Recently, customer involvement has been recognized as a means to leverage PSD (see E.G., Keil and Carmel 1995; Sawyer 2000). While its equivalent in custom IS development — user involvement — is a well established ingredient in both literature and practice (see E.G., Carmel and Sawyer 1998; Greenbaum and Kyng 1991), however, customer involvement in PSD is yet to gain momentum. Such involvement is fairly uncommon and often based upon indirect links such as intermediaries or customer surrogates (Keil and Carmel 1995).

The relative rareness of customer involvement in PSD can be associated with at least two customer involvement barriers. First, PSD is characterized by geographically distant customers (Sawyer 2000), making face-to-face interaction difficult to achieve. Second, packaged software customers are unknown (Grudin 1991) in that they are not part of any coherent use context. The development context of packaged software is clearly separated from the use context, I.E., there are no consistent use situations or organizational structures at hand when representing customers. Instead, there are multiple, sometimes conflicting, individual needs and requirements to take into consideration when developing packaged software for a mass market.

In view of these barriers, software firms increasingly develop and deploy online communities for aligning customers with their PSD processes. In such communities, customers engage in beta testing of nearly finished software as well as in more comprehensive activities including customer driven software development (Holmström 2001) and maintenance (Holmström and Fitzgerald forthcoming). In this regard, online communities can facilitate (1) the engagement of a mass scale of distributed but competent customers in developing, testing, and modifying software (Lee and Cole 2003), and (2) the development of a better comprehension of customer perspectives on the software. In other words, online communities can be understood as a means of overcoming the geographical distance and the lack of a coherent use context characterizing packaged software customers.

As documented in the literature on distributed product development, development efforts involving loosely coupled individuals and groups rely on a capability to convert the collective knowledge possessed into appropriate action for improving the product (Nambisan 2002; Orlikowski 2002). For example, in successful open source communities, this capability is a natural and important element in successful development of software (Lee and Cole 2003; Ljungberg 2000). In PSD, however, firms face a number of challenges as they attempt to benefit from customer communities. The main problem is the inherent tension between the motivational structures of commercial software firms and those of voluntary community participation. Indeed, our previous case study research of a computer game firm documents attempts to benefit from customer communities and hence, improve the development process of packaged software (Holmström, 2001; Henfridsson and Holmström, 2002). However, our research also highlights difficulties with community use in PSD, such as for example sensemaking challenges (Henfridsson and Holmström 2002) and customer role ambiguity (Holmström and Henfridsson 2002).

On the basis of a longitudinal case study at Daydream Software, this paper explores the role of online communities in PSD. The contribution of the paper is a model of community use in PSD that describes such use as three iterative and interrelated processes. These processes are unfolding at the intersection between commercial software firm practices and voluntary community participation.

The remainder of the paper is structured as follows. On the basis of a literature review, next section outlines a model for understanding how community knowledge can be built, elicited, and exploited in PSD. Then, we describe our research methodology including the research site, research design, and data sources and analysis. The next section outlines a case of community use for PSD purposes in a Swedish software firm. This is followed by a discussion of the benefits and limitations of community use in PSD. The concluding section summarizes the contribution of this paper and points out directions for further research.

2 Community knowledge in packaged software development

Developing packaged software is largely a knowledge intense activity (Clegg ET AL. 1996). Thus, it depends on the organizational capability of developing and combining market and domain knowledge into timely and competitive products (Andreu and Ciborra 1996; Prahalad and Hamel 1990). Indeed, the packaged software industry is dominated by time pressures (Sawyer 2000) in that breaking new ground is often conceived critical for generating return on investment (Zachary 1994). Contrary to custom IS development, the success of the packaged software industry's products is measured by profit and market share, underscoring the challenge of either developing a large installed base or creating new market opportunities (Sawyer 2000).

In quest for competitive advantage, software firms continuously seek new ways for improving their PSD processes (Mathiassen ET AL. 2002). Apart from efficiency measures applied to increase staff effectiveness and lower rework time (Little 2004), PSD can benefit from leveraging customer involvement and input (Keil and Carmel 1995). Indeed, many of the best ideas for product improvements come from customers (Finch 1999; Von Hippel 1986). In possessing situated knowledge of the software, customers constitute an important resource for improving PSD in that they can co-produce the software (Holmström 2001), something that Nambisan (2002) portrays as knowledge co-creation, highlighting the centrality of knowledge produced in customer—developer relationships for gaining competitive advantage over time (see also E.G., Keil and Carmel 1995).

Situated customer knowledge is a type of knowledge characterized by its contextual nature. As recognized by Brown and Duguid (2001), what individuals know always depends on — and reflects — the social context in which this knowledge is acquired and put into practice. Within the scope of this paper, situated knowledge emerges from everyday software use. Hence, it reflects not only the particular circumstances and different purposes of software use but it also conveys necessary technical competence such as hardware and software configuration skills. As such, this type of context-dependent knowledge is enacted in the moment (Orlikowski 2002), conveying the capability of individuals or groups to transform their situated knowledge into meaningful action. Such transformation is achieved through continuous reconstitution (Orlikowski 2002) and renegotiation (Wenger 1998) of meaning. Thus, situated knowledge is something that is achieved, not given, and it emerges from people's ongoing reflection, experimentation, and improvisation within the practice of which they are part (Orlikowski 2002).

Communities are mediators between individuals and formal, as well as informal, social structures, and as such, they work as repositories for the development, maintenance and reproduction of knowledge (Brown and Duguid 2001). While communities can be seen as sources of locally produced knowledge, they also create a vital link between organizational strategy and changes emerging in the surrounding environment (Brown and Duguid 2001; Lave and Wenger 1991; Wenger 1998). In similar vein, we refer to community knowledge as the situated knowledge manifested in the practice of a group of distributed software users. While a community's knowledge is not held equally by all community members but shared across the community, participation in such a community gives access to that community's identity and thus, to the collective knowledge generated within the community (Brown and Duguid 2001). Viewing community knowledge as closely related to community identity (Brown and Duguid 2001), it can be characterized as dynamic, I.E., changing as community practices changes, cumulative, I.E., expressing the community's shared history of practice, and varied, I.E., reflecting distinct skills and practices enacted by individual members of the community. In the PSD context, community knowledge is the capability of transforming, for example, design skills, graphics skills, and hardware and software configuration skills into software improvements.

In using community knowledge — as emerging within online communities — software firms seek to stimulate collective idea generation and product conceptualization among geographically distant customers (Nambisan 2002). Such use can be seen as consisting of three interrelated activities initiated by the software firm, but highly dependent on its relationship with the community. First, *knowledge building*, I.E., the process of supporting software customers' creation and sharing of situated product knowledge, is vital to improve PSD. Here, online communities work not only as enablers of knowledge exchange within communities of knowing (Boland and Tenkasi 1995), but also between the software firm and its customers. It is through dynamic interactions between such communities that new knowledge emerge (Boland and Tenkasi 1995).

Second, improving PSD cannot merely come from knowledge building. It also depends on a capability of the software firm to stimulate customers' willingness to express their knowledge and to incorporate the knowledge expressed in the PSD process. Thus, the software firm needs to develop organizational arrangements with which they are able to transform customer input into productive and meaningful improvements of the software (Henfridsson and Holmström 2002). These arrangements involve a sensemaking capability (Weick 1979) attentive to the knowledge built in light of perceived business needs. This process, presented here as *knowledge elicitation*, is understood as the process of making sense of customer generated input and transforming this into software improvements. The key to knowledge elicitation is the appreciation of shared repertoires including routines, words, tools, gestures, symbols, actions, and concepts that the community has produced or adopted in the course of its existence (Wenger 1998). Shared repertoires include both the discourse by which members create meaningful statements about the software, as well as the styles by which they express their forms of membership and ideas as community participants. The knowledge elicitation process then forms the basis for implementing useful software improvements.

Finally, *knowledge exploitation* closes the community knowledge use cycle by implementing elicited customer knowledge into software improvements. As in any product development process, this process is influenced not only by organizational factors such as firm culture, but also environmental factors including risk capital availability and market forecasts. This makes knowledge exploitation a challenge for many software firms. For example, the individualistic and entrepreneurially oriented cultural milieu of packaged software development can pose a challenge for implementing product suggestions generated by customers (Carmel 1997; Sawyer 2000).

In view of this literature review, improving PSD through community knowledge consists of three iterative and interrelated processes: knowledge building, knowledge elicitation, and knowledge exploitation. These distinctions are primarily analytical as to facilitate our understanding of the way in which community knowledge can improve PSD. Taking a cyclical perspective on community knowledge use, the community knowledge use model (see

FIGURE 1) facilitates an analysis of a software firm's repeated community activities in quest for improved packaged software. In such an analysis, the model also distinguishes the location of each of these processes. As illustrated below, knowledge building takes place within the community of software customers. While the software firm can support the initiation of this process, the building process itself unfolds through software customers' mutual engagement in a specific software product. Knowledge elicitation, on the other hand, takes place at the intersection of the community and the software firm. In this process, the software firm attempt to benefit from making sense of customer generated input in order to transform this into product improvements. To do this, there is the need for the firm to understand and appreciate the repertoire as expressed by the community as well as the community's willingness to share its inherent knowledge with external actors such as firm representatives. Finally, knowledge exploitation takes place exclusively within the software firm in order to implement customer knowledge into software improvements important for profit and market share in a competitive firm environment. In this process, the software firm faces the challenge of balancing commercial interests and customer needs and requirements.

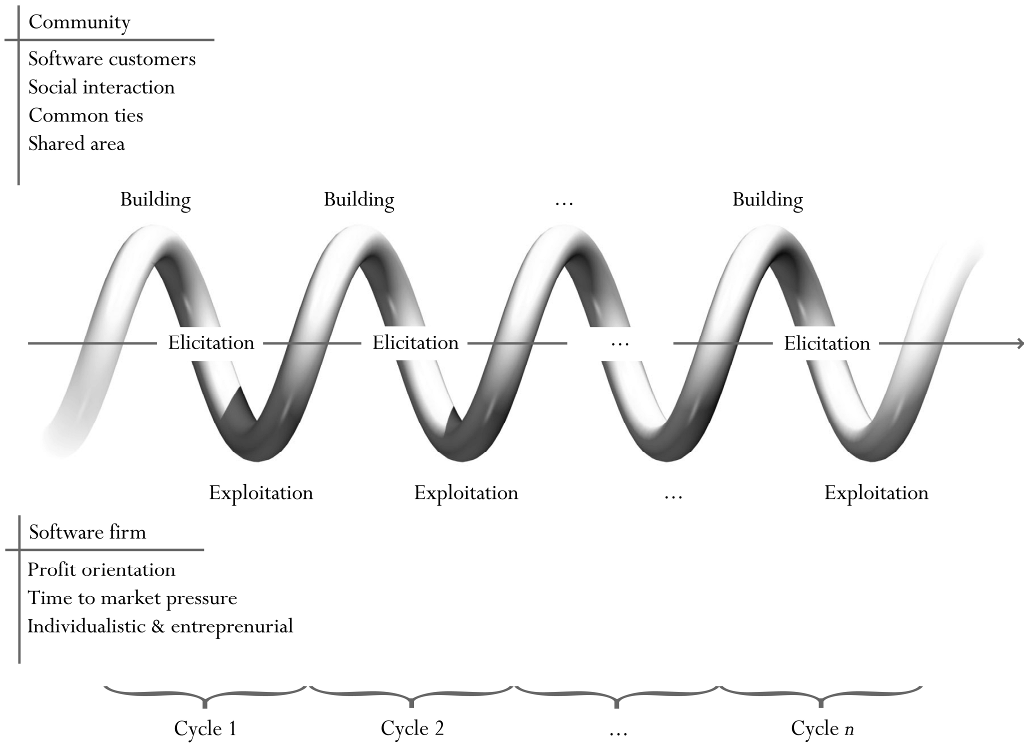


FIGURE 1. The community knowledge use cycle

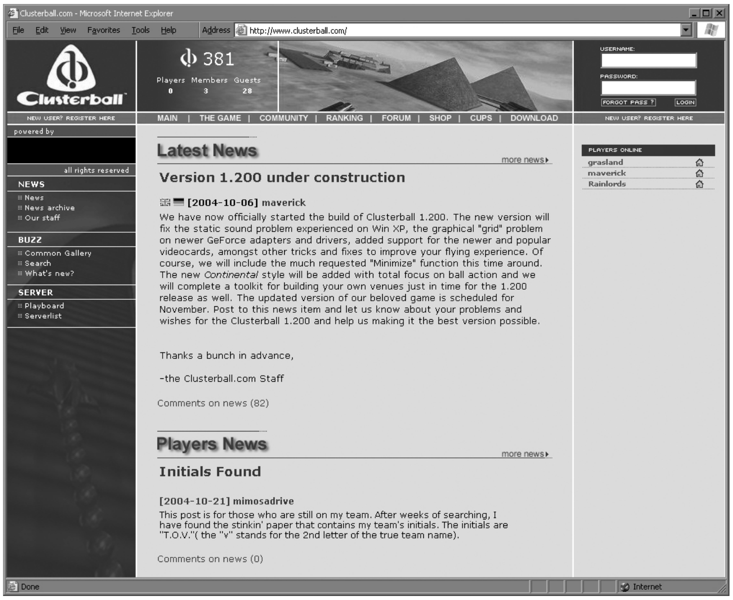
Since our own work (Henfridsson and Holmström 2002; Holmström and Henfridsson 2002; Holmström 2001) and that of others (Keil and Carmel 1995; Sawyer 2000) recognize the promise of customer involvement in PSD, we view the community knowledge use cycle as a way of focusing our thinking on the different processes of community-enabled PSD and thus, of benefits and limitations, as well as tensions, evident in these. In the following sections, we outline a case of community use in PSD in the computer game industry.

3 Research methodology

3.1 RESEARCH SITE

Daydream Software is a small computer game developer with its headquarters in Umeå, Sweden. At the beginning of this study, the firm employed around 65 people and had developed two computer games: Safecracker and Traitors Gate. The relative success of these two games had attracted a significant international customer base, making an early stock exchange quotation in 1996 possible.

Our selection of Daydream as the case for this research can be traced to a number of factors. Daydream is a software firm that focuses on a type of packaged software, computer games, that must be up-to-date with technological and societal trends. This does not only imply that software developers must be competent in a particular domain but also willing to acquire new knowledge over time. Interaction with software customers in an online customer community can therefore be seen as one source for acquiring such new knowledge. Moreover, the development, introduction and diffusion of Daydream's third computer game — Clusterball — involved a commitment to improve its PSD processes by means of an online customer community — the Clusterball community. This fact coincided well with our intents to study the role of customer communities for improving PSD.



Latest News

Version 1.200 under construction

[2004-10-06] maverick

We have now officially started the build of Clusterball 1.200. The new version will fix the static sound problem experienced on Win XP, the graphical "grid" problem on newer GeForce adapters and drivers, added support for the newer and popular videocards, amongst other tricks and fixes to improve your flying experience. Of course, we will include the much requested "Minimize" function this time around. The new *Continental* style will be added with total focus on ball action and we will complete a toolkit for building your own venues just in time for the 1.200 release as well. The updated version of our beloved game is scheduled for November. Post to this news item and let us know about your problems and wishes for the Clusterball 1.200 and help us making it the best version possible.

Thanks a bunch in advance,

-the Clusterball.com Staff

Comments on news (82)

Players News

Initials Found

[2004-10-21] mimosadrive

This post is for those who are still on my team. After weeks of searching, I have found the stinkin' paper that contains my team's initials. The initials are "T.O.V."(the "v" stands for the 2nd letter of the true team name).

Comments on news (0)

FIGURE 2. The Clusterball website

The technical basis for the Clusterball community was a web application (www.clusterball.com, see also figure 2 above) and the game itself. The web application provided electronic discussion forums, fan website links (links to unofficial Clusterball websites), product information, and software downloads. The game included a pre-game chat allowing for players to meet before each gaming session to discuss tactics and to share experiences from previous gaming sessions. Given the ambition to improve the psd process, the Clusterball community represented an attempt to cater for situated customer knowledge.

3.2 research design

The research reported in this paper builds on a 17 month (January 2000 — May 2001) interpretive case study (Klein and Myers 1999). Interpretivist researchers examine research phenomena through investigating the different meanings people assign to them (Orlikowski and Baroudi 1991). This procedure assumes that people act-in-the-world on the basis of their subjective and inter-subjective meaning creation. Whether the meanings associated with the phenomena are « correct » descriptions of the world is not an issue for the interpretive researcher. What matters is rather the extent to which these meanings can help the researcher understand why people act as they do.

The interpretive case study is particularly suited for studying research contexts in which different actor groups' views of the research phenomenon can be expected to be divergent. In fact, examining multiple interpretations of the research phenomenon is at the heart of interpretive research (Klein and Myers 1999). In our case, we suspected that the gap between Daydream's commercial interest and the non-commercial interest of community participants would be central to understand benefits and limits of community use in psd. Therefore, we reasoned that an interpretive frame of reference would be useful for exploring the prospects of online communities in psd, as these were reflected in the assumptions, beliefs, and knowledge held by both parties.

3.3 data sources and analysis

Faithful to general principles of interpretive research, the findings generated in this research have emerged as an iterative process between theoretical conceptions and empirical data (Klein and Myers 1999). As is common in most long term research, the initial conceptual apparatus — encompassing certain assumptions, beliefs, and rationale — transformed over time. In interpretive research, transitions between theoretical conceptions and empirical data are central elements in formulating an increasingly plausible understanding of the research phenomenon. In our study, such transitions (between parts such as community postings and Daydream's software development context, and wholes such as the knowing-in-practice literature) were necessary to successively formulate our understanding of community use in psd.

In this research process, a multitude of data sources have been used: meeting minutes, observational data, press releases, shareholders' prospects, technical documents, qualitative interviews, website data (community postings), and a web survey. As a significant starting point for acquiring an insider view of research phenomenon, for instance, the first author of this paper spent most of her working time during 4 months at Daydream's sites. Supporting engagement between researchers and research subjects (Nandhakumar and Jones 1997), these observational studies were important impetuses for developing a first tentative understanding of the research setting. With this pre-understanding as a basis, this paper uses three data sources from the overall Daydream study. First, qualitative interviews with Daydream employees were conducted to get an understanding of their view of the PSD process and the role that they attributed to the online community in this process. To cover the wide range of Daydream employees and their different experiences of the PSD process, a total of 11 interviews were conducted with marketing and administrative people, as well as software developers and web designers. Each interview lasted about 1½ hour and they were all tape recorded and transcribed. The open ended questions covered topics such as the development process, the role of customers, customer relationships, and customer community participation.

Second, postings published at the online community were collected between July 2000 and October 2002. These were used as an attempt to understand the way in which knowledge was acquired and exchanged among customers. This data source revealed the extent to which Daydream software developers were active in the community and in what situations there was an interaction between customers and developers. With approximately 17,000 postings divided between two forum tracks (the general track and the technical track) during this period, the Clusterball community was an active forum with postings reflecting the situated knowledge of customers important for improving Daydream's PSD process.

Third, a customer web survey were used in order to allow for distributed customers to reflect upon the PSD process and the extent to which they felt that their knowledge was built, elicited and exploited in this. To represent the wide range of customers in the community, we selected 200 Clusterball players ranging from *newbies* (not very experienced players) to *grand masters* (very experienced players) who had registered in the customer database and played the game the month before the survey was sent out (I.E. October 2000). The survey, consisting of multiple choice questions as well as free text alternatives, was distributed in November 2000 and the final answers were collected in February 2001. With a response rate of 52 percent, the survey helped us in (1) our understanding of the customers and their apprehension and application of the customer community for the purpose of knowledge building, and (2) our understanding of the customers and their view of how their knowledge was elicited and exploited by Daydream in the PSD process. However, despite a relatively high response rate indicating a rich data material, the result of the survey was a bit of a disappointment. While the multiple choice questions were carefully answered by a majority of respondents, the free text alternatives — which were intended to provide us with individual reflections in similar to those attained when conducting qualitative interviews with Daydream employees — were only briefly answered by a minority of the respondents, providing us with very limited data material reflecting the view of individual customers. As a result of this, the web survey was used only as a complement to other data sources such as observations, document reviews, community postings, and qualitative interviews.

3.4 THE CLUSTERBALL CASE

Cycle 1 Daydream's plan to develop a new computer game was formed in 1998. Since this game was envisioned as the first online game that would be distributed, paid, and played over the Internet, considerable time was invested in developing a new network protocol, micro payment functionality and a customer relationship management (CRM) application interface. The development of these new and, at that time, immature technologies was both time consuming and complex, and already at an early stage the game was delayed. At this stage, the PSD process was basically an undertaking between in-house software developers and external technology vendors and consultants.

As the first interface towards customers there was the first version of the Clusterball website in early 2000. Since the game was still under development, the website merely contained game information and development progress reports. In other words, the website was primarily an advertising tool, intended to maintain interest among earlier Daydream customers as well as to provide feedback to impatient investors in the small stock listed firm. In parallel with the development of the website, the software was beta tested by 200 players. These were recruited via the Clusterball website where customers registered in the customer database, as well as via personal contacts among Daydream employees. Approaching the official release in July 2000, the website was continuously developed to include also payment functionality and more advanced graphics and layout. At this stage, there were already hundreds of registered members in the Clusterball community.

Following the release, customers had immediate concerns with getting the software to work with their various technical configurations. Only during the first month, two software patches dealing with installation, startup and configuration problems and host errors were released. These patches were direct responses to community postings, revealing the kind of problems typical for software that suffers from tight deadlines where too little time for comprehensive testing is given. Illustrative postings from this period were:

> *"Right when the loading screen appears I get an illegal operation and I have to close it. This happens every single time. I have a* DIAMOND VIPER V550*, running* 1280 X 1024 32 BIT*, and* WIN 98*. I have had some problems with other games not switching to* DIRECT 3D *mode but nothing like this. Please help, I'm very annoyed."*

> *"Hi, My crashes end with:* CLUSTERBALL *caused an invalid page fault in module* CLUSTERBALL.EXE *at* 015F:0054E7E4*. It crashed mid-game. I have a* 450 ATHLON, 256 MB, GEEFORCE 256*. Any suggestions? Thanks."*

Cycle 2 Given the customer involvement vision, the early use of the Clusterball community came as a little disappointment. Even though the community was helpful for corrective bug fixing, this type of customer input reflected immediate customer concerns rather than the customer—developer relations envisioned in setting up the

Clusterball website. Seeking such relations, however, Daydream intensified the efforts to keep customers updated with Clusterball information to maintain and strengthen their interest. In complementing general information such as technical specifications, screenshots, and game descriptions; both software developers and marketing people published updated information including Clusterball news, technical support, and FAQ additions more frequently. Indeed, such information was considered necessary for building community knowledge. The manager at the time commented:

> *"You have to encourage the players, have a dialogue with them and make sure that they can contact us and communicate with us as well as with all the other players…this can be done in electronic forums, on chat lines or through fan websites developed by individual players and promoted by us."*

To handle the overwhelming amount of customer feedback during the month following the software release, Daydream appointed a community manager in August 2000. As a link between customers and software developers, the community manager was responsible for stimulating customer feedback as well as to make sure that this was implemented in the PSD process. This activity was considered important for building a sense of trust among customers, whose suggestions then would be better catered for, and for improving Daydream's PSD process.

The community manager viewed himself as a Daydream representative in the community, directed at encouraging customer feedback as well as for feeding customer suggestions into the PSD process:

> *"…whenever I enter the forum I do it as a Daydream employee, which is very important to remember. Thus, it is not my personal opinions — but instead the Daydream view — that I am supposed to deliver. (…) As a manager, I try to collect all the ideas and turn them into software improvements by handing them over to the developers. It is easier if this is done by one person so that the developers don't have to keep track of all ideas themselves. I know what they are doing and what people to ask for certain things and in that way I think we get a smoother and faster process in implementing new features of the game."*

In other words, the community manager played an important role in encouraging customer suggestions, making sense of these suggestions and translating their meaning into product suggestions that could be handed over to the software developers. Indeed, the community manager viewed customer suggestions as a key component in the PSD process:

> *"One very important thing in this is to remember that if you once invited the customers to be part of the development process…to make them aware of that they have impact…then you must also see to it that the suggestions they come up with are implemented. Things must happen on the basis of community discussions and it must be evident that they are able to influence the software development process in the way we told them that they would."*

Besides monitoring community postings, the community manager actively stimulated new suggestions for software improvement. Typically, the manager intervened with a clear goal in mind, E.G., a new patch release. Consider the community manager's posting in relation to the third Clusterball patch:

> *"Hi! In developing the next Clusterball patch we would like to know what features you most of all would like to see in the game…are there anything missing in the game right now and what is it that you all really want us to implement? Please, post your suggestions to the community forum…we are looking forward to seeing them."*

As a response to this invitation, numerous postings were submitted to the community. Typical examples, which later were transformed into software improvements, were:

> *"My experience of* DNF *is that I send data, but don't receive anything, and when enough time has passed, my computer evaluates this as the server having gone down (not closed, that is a different message), and ends the session."*

> *"I would like to improve the match making — to allow the possibility to find other players closer to my skill level."*

> *"I would like to be able to set a minimum and maximum player ranking when I host a game."*

> *"I would like to have the possibility to talk to other players while waiting for a game."*

The third Clusterball patch was released in mid October, 2000. At that time, the peak of corrective bug fixes seemed to have passed, and consequently, Daydream could pay more attention to functional improvements and additional features of Clusterball. In response to customer suggestions as exemplified above, additional functionality such as replay and recording, a DNF feature (players still get points even if they did not finish the game), improved match making capabilities (to lock a game server on a min/max player ranking basis), a pre-game chat, and team play ranking were implemented. The community manager noted:

> *"The first patches were developed almost exclusively on the basis of community discussions in which customer needs were evident. For example, there was the replay function so that people can record their games and look at them afterwards, the team play ranking in which every individual team gets points and are ranked in a system and a lot of different search and sorting possibilities so that it is easier to keep track of different players, different venues and different hosting servers."*

The direct link between customer suggestions and Clusterball patches was also pointed at by one of the software developers:

> *"The pre-game chat was a direct result of customer suggestions. Very soon many players felt that they wanted a place were they could meet and talk before joining a gaming session."*

As a direct customer—developer link, the community served many purposes and the community manager role was appreciated by customers. One of them expressed the following in our customer web survey:

> *"It is a good idea since we now know that someone is taking care of all the suggestions…hopefully in a systematic way. Even though they might not always be implemented anyway…but it feels better."*

Also, the overall effort by Daydream to strengthen the customer—developer link was indeed appreciated by customers:

> *"I think I can influence almost all things…Daydream has been working very closely in cooperation with the community."*

> *"I think the programmers of Clusterball will listen to peoples' voices because they really want this game to be as good as possible."*

> *"I think peoples' suggestions to new features are definitely taken into consideration and I think that's a great idea. The people at Daydream seem to be open to suggestions from us players."*

> *"The community helps me influence the people at Daydream. I talk to them there and I think they listen to us players and try to make the game as good as we want it to be."*

Cycle 3 In November 2000, Daydream appointed what they called Clusterball ambassadors as to further strengthen their relationship with customers. These ambassadors were customers playing Clusterball on a daily basis and contributing extensively to the community in terms of postings. Since the ambassadors would be regarded as any ordinary player by the rest of the community, Daydream reasoned, they could obtain information that would not be presented to, or appreciated by, Daydream employees. By the end of November 2000, there were five ambassadors — three in the United States, one in Germany and one in Italy. The ambassadors' role in knowledge building and elicitation was recognized by the community manager:

> *"Even if my job is to keep track on the community I am not an ordinary player who enters the game whenever I like to or with the same prerequisites as any other player…the ambassadors are ordinary players…in that sense I work as a link between the customers and Daydream while the ambassadors work as a link between the customers and me."*

The role of the ambassadors was further explained by the community manager:

> *"…their [the ambassadors'] duty is to be active participants in the forum discussion, help newbies in the game and see to it that people get answers on their questions when entering the Clusterball world."*

Furthermore, the ambassadors helped Daydream in administrating different community events — something that was appreciated by the community manager:

> *"To this day, the ambassadors have arranged their own tournaments and helped us in the administration of different events, they have guided new players and helped them in learning the game and so on…"*

In addition to knowledge building and elicitation activities initiated by Daydream, Daydream also sanctioned the initiative to start the Clusterball School. This initiative originated from one of the most well known and frequent players in the community. The aim of the school was to help new players to learn the game faster and thereby making them better prepared for gaming sessions. Daydream viewed the initiative as a way to attract new players to the game as well as for players to further develop their gaming skills.

The community manager appreciated the initiative:

> *"The school is great…all people seem to like it. People get to know each other without always having Daydream people around. They can learn by themselves and ask each other when they need help."*

In other words, the Clusterball School seemed to increase customer—customer links. One of Daydream's software developers noted that:

> *"I think the school helps people to get to know each other. Experienced players meet with new players and they can develop relationships that probably will last even after they have played the game. Also, I think they enjoy the game more when they know the opponents…it is good for the overall community atmosphere."*

Customer—customer links, as manifested by the Clusterball School and the community forum, were appreciated by customers too:

> *"In the community I can learn from more experienced players and from the developers."*

> *"The community is good when you need help. And it is a great way of sharing your own experiences as well as for learning from other players. I know I have learnt a lot."*

> *"It gives people a chance to give suggestions about the game, about other players and get advice for the game. I think it is a great place for newbies too, to give them an idea of what's going on and how to play the game."*

After a series of knowledge building and elicitation activities including the community manager role, the ambassador role, and the Clusterball School, Daydream could relax somewhat at the turn of the year. Accordingly, two Clusterball patches dealing with only minor technical problems were released. For example, the forth patch solved graphics and sound problems as pointed out by customers:

"I got an ATI RAGE 128 *graphics card... but when I play Clusterball, all the ships and balls are black. I need some help."*

"...the sound is still a problem in XP *with* SB LIVE *sound card."*

Following this, the fifth patch was released in February 2001. This patch — the GL setup patch — could detect whatever graphic card customers used and hence, install the latest drivers for that particular graphic card.

During the spring 2001, lots of customer suggestions on new features were posted to the community. This may be traced to the fact that at that time the game had been around for some time and there was the opportunity for players to reflect upon major changes and improvements instead of only minor bug fixes for solving immediate operability problems. In response to these postings, a sixth patch was released in April, 2001. This patch included improved joystick support (such as twist handle functions), LAN play without restriction of Internet access for host and improved artificial intelligence in the training (offline) mode. In addition, corrective development was also done. These corrections handled, E.G., a crash bug in the pre-game chat, a freeze bug when viewing replays, a throttle bug on joysticks, and a DNF bug in match history.

While the sixth patch included new functionality triggered by customer suggestions, however, there were also several suggestions that were ignored by the developers at Daydream. Examples of such suggestions were:

"Make more then just the ship playable, most other games have more than one model to choose from... Maybe there could be a model editor where players could make their own ships..."

"Could there be a viewer system so that my friends could watch other people play before they participate themselves?"

"I would like to see a password for the server when hosting a match. Setting the number of games or how long time the dedicated server should run. It would also be great if it was possible to send messages to the players when running a dedicated server."

While reflecting customer concerns, Daydream never explained why these suggestions were never put attention to. Clearly, there were limitations to what the customer community could influence in terms of development, despite the fact that they had been encouraged to participate in Daydream's PSD process. These limitations were recognized and explained by the software developers:

"There were requirements that were too costly or too technically difficult to realize. Some suggestions we simply didn't implement because we didn't agree with them...for example some of the suggestions about the gameplay we didn't like and therefore never implemented."

"The features that will influence us the most are those that players have difficulty with and that Daydream themselves are not 100 percent happy with. I don't think Daydream would be happy with changing the gameplay too much, quite correctly, because it would affect existing players, but they have shown that they want to implement suggestions and will add new modes and features to expand the game according to player suggestions without changing the playability of the game."

"Much of what was implemented we already recognized ourselves, since we of course played the game in the same way as all other players. However, we prioritized all input we got from the community and we always chose features that the majority of customers wanted and that we thought would be the most appreciated by the community."

As suggested above, there existed contradictory views on the actual role that the community played in the development process of Clusterball. While Daydream viewed the community as a resource for improving Clusterball, however, they were clearly selective in their assessment of what customer suggestions to implement.

This selectiveness was also recognized by customers. Some of the comments in our customer web survey were negative:

"I think the community can be used for reporting bugs and make small suggestions. I don't think I can influence the game itself, more like little improvements for a new patch — that is what they [Daydream] *seem to listen to."*

"The forum is made for people to chat about anything relating to Clusterball. What people do is that they post messages about what they like and dislike about the game. For Daydream — taking a look at the forum once in a while would be a good idea as it would give them ideas on how to improve the game and make it more enjoyable."

"I feel that it would be wise for Daydream to read through the user and player comments in the forums, and try to implement features based on these suggestions. Ultimately, the users will determine what they like and don't like about the game and if it meets the expectations they are looking for. If not, they will continue looking for something else."

As indicated above, our case study at Daydream pinpoints benefits as well as limitations associated with community use for improving customer involvement in PSD. Below, the implications related to these benefits and limitations are discussed.

4 Discussion

Earlier research highlights the specific firm and firm environment conditions associated with PSD (Sawyer 2000). It also outlines how these conditions make direct customer involvement difficult in PSD. Typically, customer involvement is based on indirect links such as intermediaries or customer surrogates (Keil and Carmel 1995). Given the central role that customers can play in successful product development (Von Hippel 1986; Nambisan 2002), we therefore investigated how online customer communities can be used to improve customer involvement in PSD.

On the basis of a literature review, we outline a model of community use in PSD. The model specifies three interrelated processes characterizing community knowledge use in PSD. These processes are knowledge building, knowledge elicitation and knowledge exploitation. In the previous section, we used the model for exploring community use surrounding the development of a computer game at a Swedish game developer — Daydream. TABLE 1 depicts the different cycles of community knowledge use as illustrated in the Daydream case.

Spanning between January and August 2000, the community use in CYCLE 1 was characterized by a relatively detached relationship between Daydream and its customers. Clearly, Daydream's early knowledge building activities were relatively simple actions, aimed at setting up the technological infrastructure necessary for establishing a community. Indeed, these activities were beneficial in that they provided an efficient means for collecting customer feedback on initial problems related to the software. However, the type of suggestions elicited primarily concerned installation and configuration problems experienced by customers. Reflecting immediate software operability concerns rather than commitment to software improvement, the customer suggestions posted were straight forward and unambiguous in character. In terms of knowledge elicitation, this basically meant monitoring the community forum for bug reports that could be useful for corrective maintenance of the software. While this was a rather smoothly operating community knowledge use cycle, however, it simultaneously failed to leverage other lessons learned in the community. For instance, the importance of customer—customer relationships was virtually unnoticed by Daydream and hence, no activities for supporting these were undertaken. In this regard, community use was somewhat restricted during CYCLE 1. In fact, it can be noted that without more active knowledge building and elicitation processes, the community worked as any ordinary bug reporting system.

Noting the simplistic community use in CYCLE 1, Daydream attempted to develop a richer and more enduring relationship with its customers at the end of August 2000. Indeed, CYCLE 2 (August — October 2000) was characterized by improved developer—customer relations. For example, the appointment of a community manager was a step towards taking more active part in the community's situated knowledge creation. This appointment was intended to stimulate knowledge building and sharing, as well as to improve Daydream's capability to elicit the knowledge that was built and shared among community members. As a result, the knowledge elicitation of CYCLE 2 not only included bug report monitoring but also efforts to build an insider's view of community activities. The community manager's daily participation in forum discussions and game playing contributed to such a view, making him part of the skills and practices enacted by community members (CF. Brown and Duguid 2001). At this point, customers within the community viewed the community manager's active community participation as a legitimate attempt to improve customer relations and to involve them, and the knowledge they possessed, in the improvement of the software. This sense of trust coincided with a third patch including simple corrective bug fixes but also more creative functionality additions.

Cycle	*Knowledge building*	*Knowledge elicitation*	*Knowledge exploitation*
#1: Jan—Aug 2000	Development and release of Clusterball website. Game and development progress reports. Recruitment of beta testers.	Monitoring of community postings for bug reports.	Software patches (#1 & 2) consisting of corrective bug fixes and solutions for host error problems.
#2: Aug—Oct 2000	Website updates on Clusterball news and FAQS. Appointment of community manager.	Monitoring of community postings for bug reports. Community manager participation in community activities and game playing (insider view). Community manager reports of customer needs and requirements as well as customer behavior to software developers.	Software patch (#3) including corrective bug fixes and functionality additions.
#3: Nov 2000 — May 2001	Appointment of Clusterball ambassadors. Clusterball School.	Monitoring of community postings for bug reports. Community manager participation in community activities and game playing (insider view). Community manager reports of customer needs and requirements as well as customer behavior to software developers. Clusterball ambassadors as links between customers and community manager and software firm.	Software patches (#4 & 5) including corrective bug fixes. Software patch (#6) including functionality additions.

TABLE 1. Community use in developing Clusterball

Even though the appointment of a community manager represented a step towards more advanced community use, however, Daydream took further measures to reinforce links between customers and the PSD process. In CYCLE 3 (November 2000 — May 2001), Clusterball ambassadors were recruited for improving customer—customer relations with the potential to increase the knowledge building and elicitation processes and hence, the development of the software. Complemented by the initiative to start the Clusterball School (an attempt that was sanctioned by Daydream), the ambassadors augmented earlier knowledge building and elicitation activities as conducted by the community manager. Since the capacity of the community manager was limited in terms of workload and the degree to which he could act as a true insider, these community driven knowledge building and elicitation activities targeted a broader range of community members. Thus, with the help of the ambassadors and the Clusterball School, Daydream was able to more actively stimulate customer—customer relations and also, to further strengthen the firm—community relation. Triggered by this intertwined community relationship, more extensive and far fetched suggestions were posted to the community forum. For example, suggestions on a model editor for making more than the ship playable, a viewer system and a message system for players when running a dedicated server were posted. However, none of these suggestions were implemented and hence, did not result in any improvements to the software. Indeed, CYCLE 3 demonstrated limits to community use in PSD. For instance, suggestions concerning the overall storyline were sparsely considered. Concurring with the individualistic culture of software firms (Sawyer 2000), such suggestions were rejected due to developers' strong images of what the game would be like. In addition, more costly or technically difficult changes were often rejected because management feared that they would not pay off in increased sales or resonate well with strict time-to-market deadlines. In particular, this represented a fear of not representing the majority of customers' needs and, in this way, leaving a large segment of customers outside the negotiations of intentions (CF. Schwen and Hara 2003). Despite Daydream's intimate community use, CYCLE 3 can be considered incomplete in that intensive activities for knowledge building and elicitation lead to only limited exploitation in terms of software improvements. As can be seen in TABLE 1, the patches released during this period primarily covered corrective bug fixes.

Our exploration of the role of customer communities in PSD has resulted in a model of community use in such development. While the open source literature documents successful examples of community based software development (see E.G., Lee and Cole 2003; Ljungberg 2000), there exists few studies investigating community use in commercial software development efforts. Nambisan (2002) presents an attempt to formulate a theory targeting new product development in general. To this end, Nambisan generates a set of useful propositions for understanding the theoretical underpinnings of community use. Complementing Nambisan's effort, however, our research represents a framework particularly focusing on community use in PSD. This model was furthermore used to analyze an empirical case study that was undertaken at a Swedish computer game developer. In our study, we were able to convey the challenges residing at the boundary between commercial firm interests and the voluntary nature of online community participation. Indeed, at the heart of PSD, there exists a commercial interest not directly presented in, E.G., open source software communities. In PSD, the software firm and its related online community can be seen as two separate communities-of-knowing (Brown and Duguid 2001), governed by different motivational structures and identities. In looking at the empirical data from our study, we believe that this fact has consequences for the different processes of community use in PSD.

Community knowledge building is relatively independent of what a software firm does to promote it. As long as the boundary object — the software — attracts enough committed customers there typically will be software users internalizing situated knowledge from software use, which a significant portion of them will share with others. This can be facilitated by the software firm by setting up websites and to be active in the forums established on such sites, but it is primarily something that resides naturally within a popular online community. In other words, given the existence of interesting software in the first place, knowledge building exists relatively independently of actions taken by the software firm. However, knowledge building is an important impetus to the sensemaking processes characterizing knowledge elicitation. It is therefore vital that the software firm engages in knowledge building by developing boundary spanning organizational arrangements. In the Daydream case, these consisted of the community manager role, the Clusterball ambassadors, and the firm sanctioned Clusterball School.

Community knowledge elicitation takes place at the intersection of the software firm and the online community. As long as the community is simply used as a bug reporting system, knowledge elicitation typically consists of monitoring community forums for postings about software operability problems. At a more advanced level, arrangements spanning organizational boundaries are central to internalize the customs and common ties characterizing an online community-of-knowing. These are necessary to build the customer relationship needed to acquire more substantial customer input. Given such feedback, however, the person, or group of people, acting in the boundary spanning role needs to develop a sensemaking capability for interpreting customer feedback in light of in-house development visions and ideas as well as perceived business needs. Such a capability is likely based on a dual role, grounded in the participation in — and practice of both communities-of-knowing.

Community knowledge exploitation represents the most contentious process in the community knowledge use cycle. Since the software firm operates in a commercial environment, issues related to business environment and firm culture naturally become ingredients in software improvement decisions. Partly outside the scope of what takes place in the community, commercial interests will almost always be prioritized in cases where these contradict community input. Such prioritization is typically difficult to explain to devoted community members sharing a different opinion. While such prioritizing always exist in software development projects, it can be harder to maintain trust in prioritizing based on commercial appropriateness rather than software excellence. This is a core difference between community use in PSD and, E.G., open source software development.

On a theoretical note, it must be emphasized that community use in PSD can never fully embrace the situated learning taking place in an online community-of-knowing. As highlighted by Schwen and Hara (2003), the online community literature tends to romanticize the notion of community. Looking at our case, the mere application of the notion of *use* in *community use* suggests a kind of detached relationship to the complicated and situated learning characterizing customer—customer relationships and identities produced and reproduced over time. In this regard, communities can never be controlled or exploited in conventional terms. In the spirit of Wenger's (2000) discussion on community of practice design, however, the community use model presented in this paper should be seen as an attempt to describe what actions software firms can take to recognize, support, encourage and nurture customer communities in order to improve their software development through community knowledge.

5 Conclusion

Triggered by the recognition of customers as less involved in PSD (Sawyer 2000), this paper sets out to explore the use of online communities for improving customer involvement in PSD. The paper presents a model for analyzing community use in PSD as an undertaking between a software firm and its customer community. Our model depicts community knowledge use as an iterative cycle consisting of three interrelated processes: knowledge building, knowledge elicitation, and knowledge exploitation. Also, it illuminates the relative location of these processes, pinpointing the interplay between the commercial endeavor of packaged software development and voluntary community participation.

Our analysis of the Daydream case highlights that successful completion of community knowledge use cycles can be a challenging task. As long as community knowledge is merely exploited for software operability concerns, such completion is relatively unproblematic.

Community knowledge use is then a straight forward process consisting of the setting up of a web infrastructure for stimulating, monitoring, and implementing customer suggestions. Community use in PSD is also useful for minor functionality additions. For such software improvement, cycle completion is more complex though, requiring organizational efforts for deepening the relationship with customers.

However, our study also demonstrates limitations associated with community use in PSD. These limitations are particularly true in relation to major software changes. While community members may be encouraged to devote time and effort to software improvement, they are likely to be disregarded when commercial interests are contradicted. In such cases, the omission to exploit community knowledge leaves the community knowledge use cycle uncompleted. Thus, while the processes of knowledge building and elicitation reflect altruistic ideals and mutual engagement grounded in a common interest, the process of knowledge exploitation is located within the realm of the software firm. Steered by commercial ideals, this makes vibrant community input subject to firm prioritization, typically with the result of being neglected. This tension in motivational structures is at the heart of community use in PSD.

Looking back at our study, the community knowledge use model is developed within a computer game setting. Due to the extraordinary motivation found in such communities, this limits our ability to generalize the model to other contexts. As in all interpretive case studies, our findings should be seen as tendencies rather than predictions (Walsham 1995). However, we do believe that the model and hence, the opportunity for community knowledge use, is applicable to settings in which the software concerned triggers customer motivation and engagement equivalent to that found in computer gaming. Since *"theory may never be scientifically generalized to a setting where it has not yet been empirically tested and confirmed"* (Lee and Baskerville 2003, P. 240), further research is however needed to validate and refine the model for new research settings.

References

Abts, C. (2002). COTS-Based Systems (CBS) Functional Density — A Heuristic for Better CBS Design. In *Proceedings of* COTS*-Based Software Systems*, ICCBSS 2002, Orlando, USA, February 4-6.

Andreu, R., and Ciborra, C. (1996). Organizational learning and core capabilities development: the role of IT. *Journal of Strategic Information Systems* (5), PP. 111-127.

Boland, R.J., and Tenkasi, R.V. (1995). Perspective Making and Perspective Taking in Communities of Knowing. *Organization Science*, VOL. 6(4), PP. 350-372.

Brown, J, and Duguid, P. (2001). Knowledge and Organization: A Social—Practice Perspective. *Organization Science* (12:2), PP. 198-213.

Carmel, E. (1997). American Hegemony in Packaged Software Trade and the « Culture of Software ». *The Information Society*, (13), PP. 125-142.

Carmel, E., and Sawyer, S. (1998). Packaged software development teams: what makes them different? *Information Technology and People*, (11:1), PP. 7-19.

Clegg, C.W., Waterson, P.E., and Axtell, C.M. (1996). Software development: knowledge-intensive work organizations. *Behaviour & Information Technology*, (15:4), PP. 237-249.

Finch, B.J. (1999). Internet discussions as a source for consumer product customer involvement and quality information: an exploratory study. *Journal of Operations Management*, (17), PP. 535-556.

Greenbaum, J., and Kyng, M. (EDS.) (1991). *Design at Work: Cooperative Design of Computer Systems*. Erlbaum, Hillsdale, N.J.

Grudin, J. (1991). Interactive systems: Bridging the gaps between developers and users. IEEE *Computer* (24:4), PP. 59-69.

Hagel, J., and Armstrong, A. (1997). Net Gain — expanding markets through virtual communities. *Harvard Business School Press*, Boston, MA.

Hamel, G., and Prahalad, C.K. (1994). *Competing for the future*. Harvard Business School Publishing: New York, UK.

Henfridsson, O., and Holmström, H. (2002). Developing E-commerce in Internetworked Organizations — A case of customer involvement throughout the computer gaming value chain. DATA BASE (33:4), PP. 38-50.

Holmström, H. (2001). Virtual Communities as Platforms for Product Development — an interpretive case study of Customer Involvement in Online Game Development. In *Proceedings of* ICIS (22nd International Conference on Information Systems), December 16-19, New Orleans, LA, USA.

Holmström, H., and Fitzgerald, B. (forthcoming). Using Virtual Communities for Software Maintenance. Accepted for publication in the *Journal of Organizational Computing and Electronic Commerce* — Special Issue on Collaborative Internet Applications.

Holmström, H., and Henfridsson, O. (2002). Customer Role Ambiguity in Community Management. In *Proceedings of* HICSS *35* (35th Hawaii International Conference on System Sciences), January 7-10, Big Island, Hawaii.

Humphrey, W.S. (1989). *Managing the Software Process*, Addison-Wesley, Reading, MA.

Keil, M., and Carmel, E. (1995). Customer—Developer Links in Software Development. *Communications of the* ACM, (38:5).

Kim, A.J. (2000). *Community Building on the Web*. Peachpit Press: Berkeley, CA.

Klein, H. K., and Myers, M.D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS *Quarterly* (23:1), PP. 67-93.

Lee, A.S., and Baskerville, R.L. (2003). Generalizing Generalizability in Information Systems Research. *Information Systems Research* (14:3), PP. 221-243.

Lee, G.K., and Cole, R.E. (2003). From a Firm-Based to a Community-Based Model of Knowledge-Creation: The Case of the Linux Kernel Development. *Organization Science* (14:6), PP. 633-649.

Little, T. (2004). Value Creation and Capture: A Model of the Software Development Process. IEEE *Software* (May/June), PP. 48-53.

Ljungberg, J. (2000). Open Source Movements as a Model for Organizing. *European Journal of Information Systems* (9:3), PP. 208-216.

Mathiassen, L., Pries-Heje, J., and Ngwenyama, O. (2002). *Improving Software Organizations: From Principles to Practice*, Addison-Wesley, Boston.

Nambisan, S. (2002). Designing virtual customer environments for new product development: toward a theory. *Academy of Management Review*. (27:3), PP. 392-413.

Nandhakumar, J., and Jones, M. (1997). Too close for comfort? Distance and engagement in interpretive information systems research. *Information Systems Journal* (7), PP. 109-131.

Orlikowski, W.J. (2002). Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science* (13:3), PP. 249-273.

Orlikowski, W.J., and Baroudi, J.J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information Systems Research* (2:1), PP. 1-28.

Prahalad, C.K., and Hamel, G. (1990). The Core Competency of a Corporation. *Harvard Business Review* (68:3), PP. 79-91.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems* (9), PP. 47-58.

Schwen, T.M., and Hara, N. (2003). Community of Practice: A Metaphor for Online Design? *Information Society* (19), PP. 257-270.

Von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science*, (32:7), PP. 791-805.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, (4:2), PP. 74-81.

Weick, K.E. (1979). *The Social Psychology of Organizing*, McGraw-Hill, New York.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning and Identity*. Cambridge: Cambridge University Press.

Wenger, E. (2000). Communities of Practice and Social Learning Systems. *Organization* (7:2), PP. 225-246.

Zachary, G. (1994). *Showstopper: The Breakneck Race to Create Windows* NT *and the Next Generation at Microsoft*. The Free Press, New York.