Knowledge Sharing as Spontaneous Order: On the emergence of strong and weak ties*

Karlheinz Kautz† and Morten Thanning Vendelø‡

June 28, 2001

*Revised for presentation at a Seminar on Technology Reuse and Knowledge Sharing in Software Firms, in the 2001 ROCK Seminar Series Modularity in Problem Solving and Artifact production, Department of Computer and Management Sciences, University of Trento, Italy, May 4. 2001. A prior version of the paper was presented at the International Conference on Managing Knowledge: Conversations and Critiques, April 10 - 11th. 2001 University of Leicester, United Kingdom.

†Copenhagen Business School, Department of Informatics, Howitzvej 60, DK-2000 Frederiksberg, Denmark; Fax: +45 38 15 24 01; Phone: +45 38 15 24 00; e-mail: kh.inf@cbs.dk

‡Copenhagen Business School, Department of Informatics, Howitzvej 60, DK-2000 Frederiksberg, Denmark; Fax: +45 38 15 24 01; Phone: +45 38 15 24 08; e-mail: mtv@cbs.dk

Knowledge Sharing as Spontaneous Order: On the emergence of strong and weak ties

Karlheinz Kautz and Morten Thanning Vendelø

Keywords: Knowledge sharing, networks, product development, software reuse, spontaneous order, strong and weak ties.

Introduction

In the past decade we experienced an increasing interest from both scholars in management and practitioners in the management of knowledge (Davenport et al., 1998). Also, we have witnessed an increasing interest in applying various types of IT applications to support knowledge sharing in organizations. Yet, all efforts were not equally successful, e.g., Orlikowski (1992) reported that a large consultancy company experienced severe problems in benefiting from its investment in Lotus Notes, which was supposed to allow the company to harvest from the extensive collection of knowledge possessed by its employees. In some follow ups McDermott (1999) and Steinmueller (2000) suggested that IT would do little good for knowledge management, claiming that IT can only inspire knowledge management but not deliver it.

Attempts to manage knowledge also take place in the software industry. If we look into the field of software engineering, it appears that software reuse has been a central theme for many years (Apte et al., 1990; Basili and Caldiera, 1995; Curtis et al., 1988; Karimi, 1990). As software reuse has been perceived as the key to high performance in software companies, in precisely the same way that IT-based knowledge management has been looked upon as the key to high performance in organizations. Within the field of software engineering there has been a focus on

reuse of well-defined and well-documented software objects located in software libraries (Frakes and Pole, 1994; Prieto-Diaz, 1991). However, in practice successful implementation of software libraries rarely happens.

We hypothesize that one major reason for the problems encountered with both software libraries for software reuse and IT-based knowledge sharing is that in both cases the component of tacit knowledge (Polanyi, 1962) has been overlooked. In the example of software reuse we suggest that even the best documentation of a software object cannot capture all the thoughts by its developers, and thus, tacit and probably central knowledge will not be codified. Mogens Nielsen, department manager in the Danish software company Computer Resources International Ltd., described this problem:

> "I believe that the major difficulty in software reuse concerns the problem of identifying software objects for reuse. It is difficult to describe the functionality of a software object as well as the requirements to a certain object, and then match these two in some semi-automated way and finally identify the relevant software object in the library. Before we solve this problem then I don't believe in software reuse apart from when software developers bring objects with them from one project to another."

The problem of transfer of tacit knowledge among software developers exists as it is assumed that software developers know more about their software objects than they can tell. Hence, even carefully documented software objects have undocumented features and characteristics, as object documentation solely communicates explicit knowledge transmitted into formal, systematic language. Ergo, it does not capture tacit knowledge which is of a personal kind, deeply rooted in action and involvement in a specific context, and hard to formalize and

communicate (Nonaka, 1994). In the example of IT-based knowledge management the same holds true; not all knowledge is captured and codified.
In search for a better understanding of knowledge sharing in software companies and how it evolves we studied two such companies, one was American and the other was Danish. The first company had a particular focus on software reuse, while the other had a broader view on projects, tasks and quality management as part of software product development. In both companies few or no attempts have been made to formalize the management of knowledge, yet both companies are rather successful, in contrast to what should be expected when one read the vast majority of literature on IT-based knowledge sharing and software reuse. Hence we ask:

- How did these successful patterns of software reuse emerge in these two software companies?

As our point of departure we acknowledge that knowledge networks are becoming increasingly important to knowledge sharing in organizations. As pointed out by Seufert et al., (1999, p. 180):

> "Organizations are changing more and more from well-structured and manageable systems into interwoven networks systems with blurred boundaries."

Similar observations were made by Hansen (1999), and by Zander and Kogut (1996). Hence, we know that intra organizational networks are important to knowledge sharing and transfer in organizations, and that it is possible to distinguish between strong and weak ties in networks (Granovetter, 1973). Strong ties are most effective for sharing of tacit and non-codified knowledge, whereas

weak ties are most effective for sharing of explicit and codified knowledge. Also, weak ties are more effective for identification of knowledge that might be valuable in software reuse and product development. We conceptualize knowledge sharing in the two companies as happening through strong and weak ties. Yet, our main objective is to understand how patterns of knowledge sharing emerged in the two companies studied. For that purpose we employ the concept of spontaneous order suggested by Hayek (1973), as thereby, two competing explanations; orders emerging by design and orders emerging by spontaneity are provided.
The paper proceeds as follows. First, we describe the role of networks in knowledge sharing. Thereafter we present the concept of spontaneous order. Third, we describe the methods used to study knowledge sharing in the two case companies. We present each case and analyze their patterns of knowledge sharing and how they emerged, and finally, we provide a closure.

II. The Role of Networks in Knowledge Sharing

The work presented here is among others based upon the perspective that knowledge is embedded in and constructed from social relationships as put forward by, e.g., Brown and Duguid (1991), Cook and Yannow (1993), and Nonaka (1994). The latter argue that knowledge simply cannot be processed as information, because it is continuously re-created and re-constituted through dynamic, interactive social networking activity. Knowledge is a complex, multifaceted concept. Following Polanyi (1962), Nonaka (1994) distinguish between explicit, codifiable knowledge, which can be transmitted through formal, systematic language, and tacit knowledge, which is rooted in action and difficult to formalize and communicate. In this context the traditional rhetoric about knowledge management as a phenomenon consisting of distinct phases for (gathering, collecting, capturing) constructing, distributing, storing knowledge as

put forward by Huber (1991) or Pentland (1995) is less relevant. Instead, concepts conceiving knowledge management as knowledge sharing, knowledge search and knowledge transfer becomes important (Hansen, 1999; Kautz et al., 2001). Hansen (1999) found that finding and acquiring knowledge place different demands on networks. Using the concepts of weak versus strong ties and the notion of tacit versus explicit knowledge Hansen (1999) investigated the role of weak ties in sharing knowledge across organizational sub-units. He found that weak interunit ties help a sub-unit to search for useful knowledge in other sub-units, but impede the transfer of tacit knowledge, which requires strong ties between the two parties to a transfer. Also, according to Granovetter (1973) distant and infrequent relationships, i.e., weak ties, are efficient for knowledge sharing because they give access to novel information by bridging otherwise disconnected groups and individuals in organizations. Opposite strong ties often provide redundant information, as they often exist among a small group of actors. A group in which everyone know, what the others know (Hansen, 1999, p. 83).
Turning to the concepts of tacit and explicit knowledge, then it is commonly assumed that the difficulties with and the length of the transfer go up as the tacitness of the knowledge to be transferred increases. Tacit or non-codified knowledge is best transferred through strong ties, whereas when weak ties exist among the two parties involved then transfer of tacit or non-codified knowledge is difficult. This is so because strong ties result from face-to-face interaction between two parties involved in the transfer. Also, the existence of strong ties between the parties involved in a knowledge transfer makes it more likely that they understand each other, because they share cognitive frames and hold common heuristics. One important effect of this might be that the receiver holds the absorptive capacity (Cohen & Levinthal, 1990) needed to use and benefit from the knowledge transferred. In contrast, codified knowledge is equally well transferred through weak and strong ties, and in both cases knowledge is most

easily searched for through weak ties. In total, this makes that strong ties are best for transfer of non-codified knowledge, but inhibit efficient search. Whereas weak ties impede transfer of non-codified knowledge, but provides a more advantageous search position in the network than strong ties as these ties are less likely to provide redundant knowledge (Hansen, 1999, p. 84).

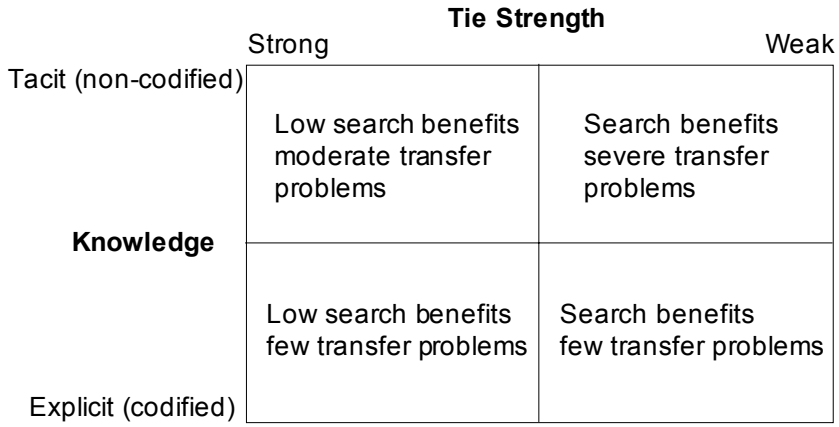

Figure 1, Search and transfer effects associated with combinations of knowledge tacitness and tie strength (Augier and Vendelø, 1999).

III. The Concept of Spontaneous Order

We now present the concept of spontaneous order and we argue why it is useful when seeking to understand the emergence of knowledge sharing patterns in software companies. The concept of spontaneous order originates from the Austrian Nobel Prize winner Friedrich A. von Hayek who borrowed it from both Adam Smith's notion of Invisible Hands and from the Scottish natural law philosophers who argued that society developed from a spontaneous order, which was the result of human action, but not human design. Hayek (1973) distinguishes between two types of order. Those made by design and those grown spontaneously, and he defines order as (p. 36):

> "… a state of affairs in which a multiplicity of elements of various kinds are so related to each other that we may learn from our acquaintance with some spatial or temporal part of the whole to form correct expectations concerning the rest, or at least expectations which have a good chance of proving correct."

Some order, consistency or constancy always exists in social life, as if this was not the case we would all face difficulties in going about our affairs or satisfying our most elementary needs. About this Hayek (1973, p. 36) says:

> "Living as members of society and dependent for the satisfaction of most of our needs on various forms of co-operation with others, we depend for the effective pursuit of our aims clearly on the correspondence of the expectations concerning the actions of others on which our plans are based with that they will really do."

Hence, as orders produces rules and institutions, such as the market, they provide for coordination among the various specialized members of society.
Designed orders "rest on a relation of command and obedience, or a hierarchical structure of the whole of society in which the will of superiors, and ultimately of some single supreme authority, determines what each individual must do" (Hayek, 1973, p. 36). We assess that much literature on knowledge management (e.g., Davenport et al., 1998; Nonaka and Konno, 2000; Seufert et al., 1999) attempts to design orders that will be efficient for companies. However, managing knowledge by design carriers the two difficulties that; a) knowledge is often difficult to articulate Hayek (1948) and structured in an imperfect way (Langlois, 1984), and b) the notion of design tends to ignore the constantly changing circumstances under which knowledge is in use, and thus, when managing knowledge by design the ever changing environment, which software companies are a part of, is neglected.

When viewing orders as emerging in a spontaneous way we subscribe to "the discovery that there exist orderly structures which are the product of the action of many men but are not the result of human design" (Hayek, 1973, p. 37). Hence, these structures are the result of evolutionary processes that nobody foresaw or designed. About the very nature of spontaneous orders Hayek (1973, p. 38) explains:

> "Spontaneous order are not necessarily complex, but unlike deliberate human arrangements, they may achieve any degree of complexity. One of our main contentions will be that very complex orders, comprising more particular facts than any brain could ascertain or manipulate, can be brought about only through forces inducing the formation of spontaneous orders."

III.i Spontaneous Order in Organizations

Hayek (1973) only described spontaneous order in relation to society, but we bring the concept into organizations, and thus, we expand the domain for the use of the concept. The legitimacy of such an expansion might be questioned, however, to our support Langlois (1992, p. 169) notes:

> "…, one finds the same sorts of unintended consequences and unplanned outcomes in the realm of organizations that one finds in spontaneous orders. … , the explanation for the existence of an organization as we observe it today is not the conscious intention of any single individual or unified group but rather the diverse intentions of many individuals and groups interacting with one another and with external circumstances over time."

Thereby, Langlois (1992) argues that it is possible to view the evolution of the firm as being spontaneous. Hence, when transferring the concept of spontaneous order to the context of organizations it can be argued that organizations are too complex to be created piece by piece in a strictly rational manner. We cannot control the particular manner in which a resulting order will manifest itself, but we may attempt to manipulate some elements of the particular circumstances of the immediate environment, and thereby, be able to initiate change of the order. Addressing the issue of deliberate organizing versus spontaneous order, Hayek, (1973, p. 46) says:

> "In any group of men of more than the smallest size, collaboration will always rest both on spontaneous order as well as on deliberate organization. There is not doubt that for many limited tasks organization is the most powerful method of effective coordination because it enables us to adapt the resulting order much more fully to our wishes, while where, because of the complexity of the circumstances to be taken into account, we must rely on the forces making for a spontaneous order, our power over the particular contents of this order is necessarily restricted."

Relying on spontaneous order rather than on deliberate organizing does in fact have the advantage that it allows organizations to benefit from the specialized knowledge possessed by the individual members of the organization. This is so because we cannot assume that any single person in an organization possesses all the knowledge needed to establish an order that will ensure the coordination of the actors in such a way that the organization benefit from that specialized knowledge of the individual members. About this Hayek (1973, p. 49) notes:

> "Organization encounters here the problem which any attempt to bring order into complex human activities meets; the organizer must wish the individuals who are to cooperate to make use of knowledge that he himself does not possess. In none but the most simple kind of organization is it conceivable that all the details of all activities are governed by a single mind."

We thereby suggest that studying knowledge sharing in organizations as resulting from spontaneous order may provide a fruitful point of departure for understanding how patterns of successful knowledge sharing evolves in organizations with massive needs for knowledge sharing. Also, we suggest that problem of knowledge sharing in organizations is essentially a problem of coordination among actors in organizations, and that spontaneous order may produce rules for coordination that allows for quick search and transfer of knowledge in organizations.

IV. Methodology

We report on case studies (Benbasat et al., 1987; Swanson & Beath, 1988; Yin, 1989) from two software organizations. In the first case the researcher spend four full days in the company observing everyday life in the company, interviewing six employees (software developers, project managers, and framework developers), collecting written documentation of the software development processes undertaken by the company for clients, and using the company's intranet to search for information. Hence, a multitude of sources of data was used. For the qualitative interviews, a structured interview-guide was developed and used to make sure that the interviews covered al aspects of interest regarding software reuse and knowledge sharing in the company. Interviews lasted between 45 minutes and 90 minutes, were taped and later transcribed.

In the second case, the research method can be characterized as action research. Action research is an approach that attempts simultaneously to achieve practical value for the client organization and contribute to theoretical knowledge (Avison et al., 1999). A number of explanations of action research cover our approach; in particular Checkland (1991). One of the authors was a member of the research team in which two researchers worked with the organization on a daily basis for two years in the period 1998–1999. A third researcher was occasionally active, mainly as a supervisor and additional adviser. The research is part of a still ongoing project and longitudinal study which aims at understanding and improving the organizations (software) work practice. An assessment and analysis of the organizations work practices was performed which resulted in an improvement plan; parts of this improvement plan have already been implemented, others are waiting for their realization. The data collected in the second case are documented as minutes from meetings, written observations and formal assessments as well as informal interviews and conversations. In addition, the engagement in the organization resulted in several presentations and reports, which served as background material for the research presented here.

V. The Case of IBIS Consulting Inc.

IBIS Consulting is a small rapidly growing IS consulting company in North California with offices in San Francisco and Sacramento. The company was established in 1994 by Scott McDonald and Brenda Wong, both coming from Price Waterhouse. Until the beginning of 1996 the company was rather small employing less than 10 consultants and the owners, but then it started growing rapidly. In 1997 it realized $12.000.000 in revenue and in spring 1998 when the study began the company employed more than 100 people. The company had been nominated as one of the 100 fastest growing companies in the Bay Area by the San Francisco

Business Times, and IBIS Consulting is a provider of Distributed Object Systems consulting to a number of different industries in the Western United States.
At the beginning of 1997 the company re-organized to support and encourage its exponential growth. New divisions were formed, each specializing in a particular technology, industry, or business process and targeting a particular type of customer. Hence, today IBIS Consulting includes five divisions: the Energy Group, the Distributed Systems Group, the I-Net Applications Group, the Electronic Commerce Group, and the Business Intelligence Group.
IBIS Consulting designs, develops and delivers enterprise-wide IT-systems for a range of large corporations. IBIS Consulting has long-term client relationship with the KAISER group and PG&E. The company's services include information systems planning; business process; application analysis, design, development and implementation; software evaluation and adaptation; and system maintenance and support.

V.i The IBIS Framework

For its development projects IBIS uses the Forte application environment and it has developed its own Forte application framework, which is used in most development projects. The Kick Start Application Framework (KSAF) is a object-oriented framework and it includes the following modules; KSAF Kernel (fundamental architecture), Asynchronous Task Management, Business Event Messaging, Cache Manager, Concurrency/Lock Management, Database Access/Persistent Data Manager, Graphical User Interface, Error Management/Exception Handling, External Application Connections, Forte Class Document generator, Logging and Diagnostics, Non-Blocking Load Balancer, Performance Monitoring, Testing and Benchmarking, Report Writing Interface, Security Management, Systems Management Agents, and Utility Classes. Also, there is a specific framework for

each of the industries that IBIS serves. All interviewees had a very positive attitude towards the framework and the benefits it provides for IBIS. Dan Tsou, a project manager, explained:

> "What makes it easy to use Forte, is that there is a recipe for doing a lot of things, If you want to do X there is a recipe for it. If you want to vary from it the approach is already there."

The Forte framework is believed to allow IBIS to complete projects faster than if the framework had not existed, sometimes cutting development time by 40-50%. Again Dan Tsou explained:

> "If we think of the impact of reuse in terms of time, then typically I think that 20-30% time is saved by using the framework. But it depends on the skills of the people involved, of course the more skilled you are the more time you can save. I think that the framework gives you the 75% to address the things that you can expect in most projects and therefore instead of coming into a project and seeing it as a new thing, you can target it with things that you have already done. So I think that the goals is not to give you a mechanism for dealing with things that you don't expect, rather it is to help you save time and energy so that you can focus on those things that you cannot anticipate."

Hence, the framework enables IBIS to complete projects by use of fewer man-hours, and thus, be more competitive. In what follows we look at how the framework emerged, how it evolves, and how it is used in IBIS.

V.ii Building the IBIS Framework

The IBIS Forte framework emerged as a consequence of major problems faced in a project that the company got involved in, as it entered a contract with Pacific Bell. Steve Simpson, Chief Technology Officer in IBIS explained:

> "We were working at Pacific Bell, in their interactive television department, and they have had about 12-15 developers working for about a year or a year and a half, before I came on board. I found a set of code, which was enormous, and I saw redundancies in every part of it. It was put together by developers from two or three different consulting companies, without any coordination. Each person was just assigned to one little problem and they were learning this new tool. In the beginning they would try to learn the tool, build a window, a screen or a call back to the services, but then as they learned more about the tool, then the second one they did they would do a little differently, and the third one as well. But the problem was that they would never go back to their first one, so we had twelve developers each with a different style, and each had the beginning style, the intermediate style and the advanced style. So I came on and saw this code base, which was so varied, and I saw them all solve the same problem in five different ways. So I started working on the abstract thing pulling things to the super-classes. First I did it at the project level, just to sort the project out. But then I saw things that we could pull out and take to any client, and so since then we have created a set of framework and training programs, and a whole set of tools that we can use to teach people the mindset. I managed to cut the code to about a third and I did it in about a week. I took the code that they had been working one for more than a year and abstracted things out. I was literally going through about 150 of these windows, cleaning code. Defining what was similar among them a moving this to super classes and so

> when you opened you would only see the differences, whereas then rest was up in the super classes. So it actually simplified the programming and maintenance became so much easier. I could not have come up with that abstraction myself of for the very first time, you almost always need to go straight to the immediate solution first, then when you want to solve them the second time, that is when you start pushing it up to the super class."

This effort resulted in the first version of KSAF. After creating the first version of KSAF it was considered how to present it in the company and get it accepted for general use. About this process Steve Simpson explained:

> "One of the things we have done is acknowledging that developers only learn if they have to solve a problem, if you tell them all the solutions to all the problems in the world, then they will forget. So the biggest thing is that in our training we give them a problem to solve. When I introduced the framework in IBIS, I had to find a way of showing people where things belonged, so I actually said exercise no. 1: build a simple window with an OK bottom, using the framework, then exercise no. 2 and so on. So I lead them through the course using more and more of the functionality. In total I made about eight exercises. For a junior developer it might have taken a couple of days, but a senior person would finish the whole thing in about two hours. But the key is that you have to give them a problem to solve and give them half the solution, and then they will figure it out themselves. So the thing is that you cannot just drop a manual."

Over time this set of exercises evolved into what is now known as the Kick Start Development Methodology, which is taught to all new recruits in IBIS, and sold to customers who want to use KSAF in their own systems development activities. As

new recruits enter IBIS they go through a week long course training them in the use of KSAF. Steve Simpson explained:

> "We use the kick start methodology internally a lot because it is how we train people. Because it allows us to get them up to speed more quickly."

During this week the recruits develop small applications supervised by an experienced KSAF systems developer. In addition, to this introductory course, IBIS also provides a more advanced course where people are taught how to architect applications. Thereby, the Kick Start Development Methodology train people in object oriented application development, which is an integrated part of the company's use of KSAF, as the methodology focus on; early deliverables, frequent user feedback and early detection on flaws.
As KSAF constantly evolves there was also a need for constantly keeping IBIS systems developers up to date with the new modules added to the framework. A key ingredient in this process is what has become known as the brown bag lunches. Steve Simpson explained:

> "You hear about all these knowledge sharing tools and things like that, but it really a kind of comes down to people. You can put in all the tools and facilities, but if the programmers are driven by project managers and deadlines, you almost have to come to them and say I have this great brown bag lunch, I'm doing a little talk, or come in, and have pizza. I always bring in some pizza, and say come on in and share technically."

And Jim Stearley, senior systems developer, elaborated:

> "We usually set up the workshops as a brown bag lunch and then we will sit down with a lab-top and plug it into one of the screens that goes into the wall, and then we will walk through demonstrations or the code, how it works, what are the pieces involved, this is how it can be used, and this is how I use it. Steve is constantly trying to bring people, especially the news ones that we bring in, up to date. So he is giving seminars to them we have topics meetings, where we try to bring each other up to date, it is a kind of knowledge sharing thing where people would say: "You know I've tried to something like it, and I think that it can be used or done in this way." So it is a very interactive thing.

Hence, over time a whole set of different ways for making developers acquainted with the expanding framework evolved. In projects senior people supervise junior people, and help them extend their use of the framework. Show cases are made in smaller groups where people show how they solve specific problems. This diversity of knowledge sharing was developed as it was acknowledged that some developers just want to sit in the back of the room at get an explanation presented on a blackboard. Others want to be more interactive and ask questions, why did you do this? How did it work? Others want to get the code themselves and see how peers have solved certain problems, and then ask questions to the person who developed it. So all three forms are available, as people have different demands for knowledge acquisition – teaching – sharing – inquiry.
Within the organization it is also a concern that the organization does not settle with what it has and start taking established procedures for granted. Steve Simpson phrased this concern in the following way:

> "… simultaneously you need to make sure that it doesn't get too institutionalized, in the sense the people are pulled in every second week. You

need to keep it ad hoc, you can tune it to an audience, which are there at the specific point in time. You have to get people when they are ready for it."

The various knowledge sharing techniques developed and used in the organization however, have not eliminated the need for written documentation of the different parts of the framework. Hence, for all new modules added to the framework written documentation is produced. Jim Stearley explained:

"I wrote some 70 pages of documentation. We have basic framework documentation that gives you the class name it inheritances each attribute is described, each method is described and what it does. And also we have that for each of the framework elements. The documentation is pretty good, it explains systematically what the functions are doing. This is what have to be done to set the system up, and these are the things that it usually can do."

In addition, to the initiatives described above a discussion area called the Forte Bulletin Board was launched on the IBIS intranet in March 1998. Yet, in September 1998 only two questions had been posted, and no answers had been given. Dan Tsou explained that in his opinion the Forte forum was rarely used, and he hasn't even looked at it.

V.iii Extending the Framework

As it somehow appears form above the framework is an ever expanding entity, as new modules are constantly added to it. Ideas for new modules emerge in different ways, but the process described below by Jim Stearley is very common in the company:

> "… defining new objects typically starts with, the idea that there has to be a better way of doing things, and from this idea people try to implement it in a better way, than they think it is done for the time being. I saw that people would basically copy and rewrite windows, so I developed a standard window bottom bar, and then you just choose the kind of bottom that you need, and it then handles all the things that needs to be handled. Make sure that it calls to the parent window, calls the database. So you don't have to make sure that you copied everything, and it also minimizes the risks of producing errors, that later on needs to be corrected."

Also, new modules mat be added as generic modules are extracted from projects as they are completed. Steve Simpson explained:

> "From each project we try to figure out what to abstract and add to the IBIS framework, so that it constantly gets updated with what is perceived as valuable functionalities."

Hence, expansion of the framework is very much driven by both individual employees observing what they perceive as needs for new modules, and by a routine for extraction of generic modules from completed projects.

V.iv Organizing Projects

As IBIS grow rather quickly fast learning and training of new developers is needed. Hence, the usual six months adaptation of new employees represents an obstacle to the growth of the company. To overcome this problem a certain way of organizing projects has evolved, thereby making sure that new developers can be

trained by senior people and then later on act as mentors for other new developers. Steve Simpson described it this way:

> "You build a team, and then as they become skilled we split them so that we can have the benefit for seniors working together with juniors who then get trained by the skilled developers. You build a core group and then you split it. New cores are build and then you split them again."

We may refer to this as cloning of projects, as IBIS take a senior member of a core group and put the senior into a team of new people who then work with the senior. Combined with the Kick Start training program this allows for quick integration of new recruits into project teams. Jim Stearley explained about his own experience as new employee in IBIS:

> "In my second week here I was on the project, because the Kick Start training in the company did not cover a lot of the things being used in the project. I didn't consider it to be a bad thing, it was just the way it had to be. It was a big project so there was some IBIS people there with very deep KSAF knowledge that you could go to, and I did, and I asked a lot of questions."

Development projects in IBIS all follow a specific structure, which has proved successful over time, and which is also a part of the Kick Start Development Methodology. Dan Tsou explained:

> "In each project you have basically these four different types of roles. There is the project manager, which interacts with the client and managers the deliverables, time lines, and the team. The technical leader that especially helps the consultant during the time of construction, and helps managing what is

expected to be delivered. And then there is the architect role, which is mostly involved in the beginning of the project, and as needed throughout the rest of the project. And then finally there are the consultants which do most of the work on the project."

The architects are perceived to be the key people in the company, they come up with new routines and how to accomplish new tasks, and they diffuse knowledge about it to other people in the company. In addition, the architects are the owner of KSAF and the project managers gives feedback to them about what kind of modules it would be nice to have included in the framework, but they will make the decision. So project managers are a kind of resource for ideas for how to expand the framework, and what might be beneficial in the future, and what has worked well and what has not worked well. Dan Tsou explains that he himself have suggested inclusion of functionalities which have then been included in the framework.

The overall project organization a kind of cloning of project takes place. Meaning that experienced systems developers are removed from their old team and transferred to new teams with junior people. This gives the possibility for having knowledge transferred among people in projects, as senior people mentor junior people in projects. Also, in this way recruits are socialized into the IBIS way of thinking. Steve Simpson explained:

"If you go into projects with more than 2-3 developers, you need a senior person who knows what is going on, it is too dangerous to put a bunch of junior people into it. What is nice in the IBIS framework is that for each project, you can create a project specific framework by adding a level in the framework, so that you get a project framework. In which you can make all the subclasses needed for that specific project or customer. Now the developers get all the

functionalities from the IBIS framework for free, but they don't need to know if they get them because they are in the IBIS or in the project framework, they are just there. Therefore, they can use the framework in the way they prefer, and are not restricted to use the components from the IBIS framework. Thereby, framework building becomes a problem driven process."

Hence, variation in use of KSAF constantly evolves, because in any project the framework is used in different ways and thereby, systems developers also acquire different experiences. So reuse is both an individual issue as well as an organizational issue, because it is about making the software developers having a certain attitude to their work.

V.v. Order and Knowledge Networks in IBIS

Going through the IBIS case it appears that several evolving processes, and thereby, knowledge sharing patterns contributed to software reuse in the company. First, interpreting the process as one of design or one of spontaneity, it appears that no attempts to plan and design a specific knowledge sharing system took place in the company. Instead, it appears that first and foremost the idea for what later became KSAF, emerged as Steve Simpson encountered the problem with the huge amount and very varied code produced in the project for Pacific Bell. Responding to what he perceived to be a problem, he used his extensive knowledge about the Forte Application Environment to develop a structure for the project. This step initiated an evolutionary development of KSAF, as it has never been driven by a plan, but instead by ideas conceptualized by various people in IBIS, and by taking modules developed in projects and making them generic, so that they could be used in future projects. Hence, over time rules for the expansion of KSAF evolved. The same observation can be made for the Kick Start Development

Methodology, which can be said to constitute one of the core elements of the software reuse and knowledge sharing process. In concert with KSAF the Kick Start Development Methodology provides the structure for the cognitive categories that people uses in the programming work. Hence, by use making new recruits subject to the Kick Start training program, IBIS trains them in the cognitive categories, which constitutes the basics for being capable to work within KSAF. Also, in this training strong ties are initiated with the senior person responsible for the training, however, as recruits join projects and collect experience, these ties become weaker. Yet, they still provide access to knowledge that may be needed sometime in the future.

Another order in IBIS is the overall project organization. A special feature of it, is the cloning of projects. As this cloning happens knowledge networks of strong ties, which have been established while the project team has been working together, are slowly dissolved. Because as former teammates become members of new teams, they get new experiences from work with dissimilar projects, and thus, the formerly overlapping cognitive categories become less overlapping. Also, as the former teammates build up strong ties with new teammates, they have less time to maintain their strong ties with former teammates. Hence, transformation of strong ties to weak ties happens due to two mechanisms. Again this overall project organization was not planned or designed. Instead, it foremost evolved as the company had to keep up with it high growth rate, and thus, had to find a way to increase productivity.

VI. The Case of NP

The company NP is developing an IT-based solution for the problem of how to present 20000 warehouse items on a 14'' screen through a slow network in a fast and structured way. The company was founded in 1997 by a business-man and an

IT expert. From the start of the improvement project in early 1998, the number of employees has grown from 25 to well over 60 in mid 1999. Most of the staff is directly involved in software development and the rest in business administration, marketing and sales. The two founders function as management. Software development is roughly divided into two divisions. A production division delivers standard intra- and internet solutions. Besides software developers, the division consists of operational - network administration - staff and art designers. The other part of the organization's operation takes place in the growing research and development division. It deals with *the* future product of the organization. It is this division that is the focus of this study.

NP is characterized by an entrepreneurial and frank atmosphere. There is a high degree of internal communication and social interaction, also outside office hours, and mutual adjustment is used as the primary co-ordination mechanism. Employees trust each other and show mutual respect for each other's capabilities. NP is geared by and towards innovation, and a creative, dynamic, energetic and at times hectic atmosphere prevails. All employees are placed in an open office landscape where the employees can develop a feeling of community and security and NP largely employs young, newly graduated and highly educated people who have a background in information systems, computer science or organization science and are open to newfangled ideas.

Management takes the strategic decisions on business and products. Their management style is management by walking around. Management expects everyone to assume responsibility and work for a common interest. Three project managers and a project coordinator maintain an overview of the technical tasks and achievements. The software developers enjoy a fair amount of autonomy, which extends to the solutions of technical problems as well as the purchase of tools and other materials. They have the opportunity to specialize within their fields of interest and are in many respects free to determine which tasks they would like to

concentrate on and which problems they would like to solve. They are enthusiastic about their work, and long working hours, although not necessarily appreciated, are just one sign indicating their commitment and motivation.

The work in this division is rather project and team-oriented. Three overall project groups which aim at developing the product are managed by three project managers, who function as coordinators for a varying number of subprojects where the employees, usually in groups of 4 – 5 persons, perform all sorts of development tasks. In addition, one of the project managers holds responsibility for the improvement project. These project groups frequently emerge on an ad hoc basis and are determined by the situations. Everything is performed in form of more or less formally defined projects. More as an exception than as a rule, sometimes contracts are defined with a preliminary project plan, the determined resources for the project, and the expected deliverables. Sub projects are typically organized as working groups about a specific topic and/or to solve one specific problem on a short time temporal basis. Often there are no strict boundaries between the subprojects, and the project groups only exist for a short period of time until the problem that initiated them has been solved. To solve the problems will normally involve members, especially art designers, from the production group and other subprojects. In addition, there is a general project coordinator who keeps track of the different activities and who together with the three project managers forms a so-called steering group.

All projects are however approved by the steering board or a project leader. Thus spontaneous order is achieved on the basis of ‘encouraged self organization’ within a given flexible frame of rules and procedures. The ‘emerging’ sub projects’ recruit its members in different ways:

- members find themselves on the basis of ‘shared’ interests - the initiator makes contact with people s/he wants on the team

- hear say leads to people joining the group
- people considered knowledgeable are assigned to the project -upon or without request

Yet, the bottom line is project and resource approval through a project leader or the steering committee.

Knowledge sharing in project groups and teams internally or with other knowledgeable employees takes place through numerous formal and informal, a mix of scheduled and spontaneous, meetings and conversations with company members with the respective knowledge.

Relatively little written documentation exists in the organization. This is at times perceived by staff and management as a problem and was one reason to start the improvement project.

VI.i Project Planning and Task Management as Spontaneously Emerging Orders

In a dynamic environment, new opportunities and challenges to improve work practices and the product constantly emerge, giving rise to frequent changes in requirements. Therefore, project planning and change, or task management in the case of NP, plays an important role.

Knowledge capture in relation to product amendments is generated through market research and uptake of new research results, including product ideas that emerge from the daily practice. This process is implicitly supported by IT as the Internet is surfed for interesting information, but this is not the kind of support structure we focus on here. Everyone in the organization is involved in this type of knowledge process, but it is the steering group who has the overall responsibility for managing new requirements and tasks. Before the steering group or a project leader approves

new ideas, they are stored and distributed to all interested and affected employees through an IT-based list of ideas. A crucial role in the process is played by co-ordination meetings where the steering group or parts of it and the developers evaluate and interpret the information available and decide whether to continue to explore and follow up on an idea or a new requirement. In this process, an experienced developer makes a rough estimate of the resulting task, and often he or someone else qualified performs a preliminary risk analysis. If the proposals are approved, the tasks will be subdivided and distributed.

Upon approval, an idea becomes a task, and the developers who volunteered or were assigned to the job transfer the idea description to their own planning tools. From now on, the individual developers, who constitute a small subproject, are by and large responsible for detailed planning, tracking and overview. In general, the developers perform their tasks individually, but with assistance from colleagues, if necessary. Only once a week, does the project leader, one of the three overall project managers in whose area the subproject falls, produce a status report from the information provided by the developers, and, if necessary, he will eventually take action. In the organization, this process is called participative project management.

As part of their daily work, the developers describe their subtasks based on their shared knowledge from the coordination meeting. They start by outlining their perception of the co-ordination meeting, explaining the background of the tasks and their decisions and classifying the tasks according to a simple scheme. Using a rudimentary method, the developers estimate their subtasks according to a best case, worse case and the most likely scenario in terms of calendar days and working hours. They also describe the risk and finally provide a list of all other developers involved in or affected by their actions. This can be regarded as part of knowledge construction and stored in the IT-based tool. During task performance, the developers update the information regularly. The information is *accessible* to everyone while there is constant communication within the project group and

beyond about the issue at hand. The information as interpreted by the different developers and the project leaders is used for co-ordination purposes on an informal day-to-day basis and on more formal status and co-ordination meetings.

The tool only supports distribution and storage of information to enable everyone to get an overview of the status of their own tasks, the status of others' tasks, the relation between these and the status of the project as a whole. In addition, it is used for resource tracking based on estimates and actual progress, thus supporting re-planning.

The co-ordination meetings have various effects

A new meeting structure and meeting conduction rules have been commonly developed in the for the organization usual way (see later) They support the sharing of knowledge and encouraged to engage in individual and collective reflection. This communal process strengthens the mutual commitment to fulfill accepted tasks. The developers also report that – although the IT support is quite basic – access to information, as a basis for the other knowledge processes, has been extremely simplified and improved. Estimations –available as stored numbers with additional descriptions – had been practiced and their accuracy has increased.

The descriptions and measurements are distributed as information – at best explicit knowledge – in the form of reports and graphs, which makes the information visible and accessible to everyone in the organization. The results are not only distributed electronically, but also prominently exposed on the walls of a meeting room.

As a consequence, the co-ordination meetings lead to better, more precise, individual and communal planning.

In terms of project planning and task management as knowledge (sharing) processes, we may interpret the lists of ideas and task descriptions with information about the actual course of a project as combining explicit knowledge. The explicit knowledge is internalized when used in practice to amend plans and actions. It is

externalized through the continuous dialogue between employees about the planning practice, which takes place in an open atmosphere and environment where – although tasks are generally performed individually – joint planing in the form of participative project management prevails. (This process can be seen as socialization). The only role IT support played in this connection was to provide easy access to information visualizing the facts.

VI.ii Strong and Weak Ties (and Spontaneous Order) in the SPI Project

As an example for the strong and weak ties (and a further example for the emergence) of spontaneous order the establishment of the whole SPI project and some of its subprojects shall be used now as an illustration

Through another co-operation project management was in loose contact with an University research group which works with process improvement. Upon a formal request from management the leader of this research group scanned his co-workers who interest and resources for such a project and the graduate students who had participated in a course on the topic and who were searching for a master project. Thus a meeting with management, the head of the research group, a senior researcher to whom he has a close relationship and two junior researchers who were close co-operators were held were in organization was provided with the basics of process improvement. In that meeting also one project leader from NP with comparably long experience in software development and highly respected by all employees, both those in his project group and all other participated.

As a result the project leader and the two junior researchers were assigned and formed the kernel of the SPI project. They were to work together on a daily basis for an initial 10 weeks period. The other meeting members took on roles as

peripheral consultants – for the part of the university members – and decision-makers for the NP leaders.

Within a short time period the two junior researchers supported by the project leader formally assessed the organization's software capability by interviewing and interacting with almost all employees in close fashion where they would listen to the employees description of the organization's strengths and weaknesses and their improvement proposals. This approach got the senior researchers acquainted with the employees and led to an assessment report.

The results of the work were presented by the SPI project members to all employees again at a meeting. As the most acute problem they had identified the lack of a 'professional meeting culture' and the employees agreed on this finding. Therefore immediately after the presentation a technical working group was founded with the approval of management to solve the problem. (In accordance with the organization's work routines) half of the meeting participants were in line with their own preferences, interests and relationship to other employees assigned to that temporary working group. The other half was assigned to a different group working with a different problem. The SPI project group members scheduled the date for the first working group meeting and appointed one person, whom they had identified as very interested in the topic during their interviews, as responsible for the preparation of that meeting. They also participated in the first meeting. The 'meeting culture' group needed two more sessions to develop a solution, which afterwards on a further meeting was presented to all the other employees. It was accepted and is now implemented and the group, which had worked together for one intensive working week was dissolved again.

The assessment work and the implementation had however also created awareness and directed the SPI group to another area, namely the general area of knowledge sharing. First of all the status of the two junior researchers was changed; they were hired as full time staff without a time limitation by management to work with

improvement and two new subprojects were initiated. Let us therefore look at these subprojects as the final example (with regard to the effect of weak and strong ties on knowledge sharing).

The first subproject started was a thorough investigation of the concept of knowledge. The two junior researchers comprised the kernel of the project group. They had developed a close relationship to their university senior partner and a further, less close link to another lecturer of a different group. They used both these informants and the participation in a university course to learn about the term knowledge. They read new literature and compared these sources by discussing both with their supervisor, the project manager who was member of the SPI group, collected further information from top management and included several employees whom they knew from the earlier assessment, their daily work and earlier improvement actions in their investigation. The results were documented in a report and were frequently presented to the organization. As a consequence a further subproject emerged and was approved with the aim to develop IT support for collecting and sharing knowledge about project management (see sec. VI.i). This subproject consisted of the two junior researchers, a newly hired developer from the researchers wider university network – he was a specialist for tool development – and a further tool specialist from the university research team leader's network. Again the management and the project manager were regularly included when requirements for the tool needed to be approved, while the juniors' university supervisor and two further university senior staff – the research group head ands one of his closest co-workers participated in evaluating several of the evolutionary developed tool prototypes.

Being, SPI, and thus among others project planning specialists, the junior researchers continuously extended their information about project planning again by working with all project leaders and employees who were skillful with respect to

task management – they had also been identified through the assessment and the earlier improvement work.

Last, but not least, the junior researchers frequently asked for advice when it came to the technical aspects of the tool. Here they relied on the knowledge of several of the company's specialists in design of user interface, database architectures and access and search engine development. These employees provided their knowledge either demand or in informal conversations. Although they were not formal members of the subproject, they played a significant role in the project group's work and success. After completion and introduction of the tool, the project group was dissolved and the SPI group concentrated itself on new tasks.

VII. Closure

In both IBIS and NP knowledge is kept floating in different albeit effective ways, which, however, cannot be described as similar. Yet, the processes by which the patterns of knowledge sharing evolved in the two companies exhibit the quality of being spontaneous orders. Hence, we suggest that there is reason to believe that often patterns of knowledge sharing are not designed. Instead they emerge in spontaneous ways.

Another interesting observation from the two cases is that IT plays a subordinate role in knowledge sharing. IT may provide assistance for visualization and access of information as a basis for knowledge sharing, but the interpretation of such information is a human activity that cannot be assisted directly. IT can only, if at all, assist implicitly in providing information to support the processes and circumstances that enable knowledge creation and knowledge management.

Acknowledgements

Chrysostomos Mantzavinos, Department of Political Science, Stanford University guided us in our adventure into the concept of spontaneous order. Part of the empirical research in IBIS Consulting Inc. was undertaken in spring 1998 while the second author was a Fulbright Research Scholar at Stanford University.

References

Apte, U., Sankar, C. S., Thakur, M., and Turner, J. (1990) Reusability Strategy for Development of Information Systems: Implementation Experience in a Bank. *MIS Quarterly, 14*, pp. 421-431.

Augier, M., and Vendelø, M. T. (1999) Networks, Cognition and Management of Tacit Knowledge. *Journal of Knowledge Management, vol. 3, no. 4*, pp. 252-261.

Avison, D., Lau, F., Myers, M. D., and Nielsen, P. A. (1999) Action Research. *Communications of the ACM, vol. 42, no. 1*, pp. 94-97

Basili, V. R., and Caldiera, G. (1995) Improve Software Quality by Reusing Knowledge and Experience. *Sloan Management Review, vol. 37, no. 1*, pp. 55-64.

Benbasat, I., Goldstein, D. K., and Mead, M. (1987) The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, *11*: 369-386.

Brown, J. S., and Duguid, P. (1991) Organizational Learning and Communities of Practice: Towards a Unified View of Working, Learning, and Innovation. *Organization Science, vol. 2, no. 1*, pp. 40-57.

Checkland, P., (1991). From Framework through Experience to Learning: The Essential Nature of Action Research. In: H. - E. Nissen et al., (eds.) *Information Systems Research: Contemporary Approaches & Emergent Traditions*. Amsterdam: North-Holland.

Cohen, W. M., and Levinthal, D. A. (1990) Absorptive capacity: A new perspective on learning and innovation. *Administrative Science Quarterly, vol. 35, no. 1,* pp. 128-152.

Cook, S. D. N., and Yanow, D. (1993) Culture and Organizational Learning. *Journal of Management Inquiry, vol. 2, no. 4,* pp. 373-390.

Curtis, B., Krasner, H., and Iscoe, N. (1988) A Field Study of the Software Design Process for Large Systems. *Communications of the ACM, 31,* pp. 1268-1287.

Davenport, T. H., De Long. D. W., and Beers, M. C. (1998) Successful Knowledge Management Projects. *Sloan Management Review, vol. 39, no. 2,* pp. 43-57.

Frakes, W. B., and Pole, T. (1994) An Empirical Study of Representation Methods for Reusable Software Components. *IEEE Transaction of Software Engineering, vol. 20,* no. 8, pp. 617-620.

Granovetter, M. S. (1973) The Strength of Weak Ties. *American Journal of Sociology, vol. 78, no. 6,* pp. 1360-1380.

Hansen, M. T. (1999) The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organizational Sub-units. *Administrative Science Quarterly, vol. 44, no. 1,* pp. 82-111.

Hayek, F. A. (1948) Economics and Knowledge. In: F. A. Hayek (ed.) *Individualism and Economic Order*. Chicago, IL: University of Chicago Press, pp. 33-56.

Hayek, F. A. v. (1973) *Law, Legislation, and Liberty, vol. 1: Rules and Order*. Chicago, IL: University of Chicago Press.

Huber, G. P. (1991) Organizational Learning: The Contributing Processes and the Literatures. *Organization Science, vol. 2, no. 1,* pp. 88-105.

Karimi, J. (1990) An Asset-Based Systems Development Approach to Software Reusability. *MIS Quarterly, 14*: 179-198.

Kautz, K. and Thaysen, K. (2001) Knowledge, Learning and IT Support in a Small Software Company. In: *Proceedings of the European Conference on Information Systems*, Bled, Slovenia, June.

Mosakowski, E. (1997) Strategy Making under Causal Ambiguity: Conceptual Issues and Empirical Evidence. *Organization Science, vol. 8, no. 4*, pp. 414-442.

Langlois, R. N. (1984) Internal Organization in a Dynamic Context: Some Theoretical Considerations. In: M. Jussawalla and H. Ebenfield (eds.) *Communication and Information Economics: New Perspectives*. Amsterdam: North-Holland.

Langlois, R. N. (1992) Orders and Organizations: Toward an Austrian theory of social institutions. In: B. J. Caldwell and S. Boehm (eds.) *Austrian economics: Tensions and new directions*. Boston, MA: Kluwer Academic Publishers, pp. 165-183.

McDermott, R. (1999) Why Information Technology Inspired but Cannot Deliver Knowledge Management. *California Management Review, vol. 41, no. 4*, pp. 103-117.

Nonaka, I. (1994) A Dynamic Theory of Organizational Knowledge Creation. *Organization Science, vol. 5*, pp. 14-37.

Nonaka, I., and Konno, N. (1998) The Concept of "Ba": Building Foundation for Knowledge Creation. *California Management Review, vol. 40, no. 3*, pp. 40-54.

Orlikowski, W. J. (1992) Learning from Notes: Organizational Issues in Groupware Implementation. In: *Proceedings for Conference on Computer Supported Cooperative Work*. November 1. - 4,. Toronto, Canada, pp. 362-369.

Pentland, B. T. (1995), Information Systems and Organizational Learning: The Social Epistemology of Organizational Knowledge Systems. *Accounting, Management & Information Technology, vol. 5, no.1*, pp.1-21

Polanyi, M. (1962) *Personal Knowledge: Towards a Post-Critical Philosophy*. Chicago: Chicago University Press.

Prieto-Diaz, R. (1991) Implementing Faceted Classification for Software Reuse. *Communication of the ACM, vol. 34*, pp. 88-97.

Reed, R., and DeFillippi, R. J. (1990) Causal Ambiguity, Barriers to Imitation, and Sustainable Competitive Advantage. *Academy of Management Review, vol. 15, no. 1*, pp. 88-102.

Seufert, A., von Krogh, G., and Bach, A. (1999) Towards Knowledge Networking. *Journal of Knowledge Management, vol. 3, no. 3*, pp. 180-190.

Steinmueller, W. E. (2000) Will New Information and Communication Technologies Improve the Codification of Knowledge. *Industrial and Corporate Change, vol. 9, no. 2*, pp. 361-376.

Swanson, E. B., and Beath, C. M. (1988) The Use of Case Study Research on Software Management Research. *Journal of Systems and Software, 8*: 63-71.

Yin, R. K. (1989) *Case Study Research - Design and Methods*. Thousand Oaks, CA: Sage.

Zander, U., and Kogut, B. (1995) Knowledge and speed of the transfer and imitation of organizational capabilities: An empirical test. *Organization Science, vol. 6, no. 1*, pp. 76-92.