2006

Knowledge creation as an ISD goal: an approach based on communities of practice

Andrea Carugati
IESEG School of Management, a.carugati@ieseg.fr

Francesco Bolici
Cassino University, francesco.bolici@eco.unicas.it

KNOWLEDGE CREATION AS AN ISD GOAL: AN APPROACH BASED ON COMMUNITIES OF PRACTICE

Andrea Carugati, *IESEG School of Management, Lille, France,* a.carugati@ieseg.fr

Francesco Bolici, *CeRSI, Università Luiss Guido Carli, Roma Italy (fbolici@luiss.it); OrgLab, Dip. Impresa Ambiente e Management, Università degli Studi di Cassino, (*francesco.bolici@eco.unicas.it*); IESEG School of Management, Lille, France(* f.bolici@ieseg.fr)

Abstract

In this article we present a perspective that considers information systems development as a knowledge creation process. By analysing the literature we evidenced a gap between two streams of thinking: the artefact-centric and the process-centric point of view. The first one focuses on the artefact characteristics but does not consider information systems development (ISD) as a process. The second one emphasizes the development methodology and its dynamics but, even when it considers the IT artefact as the main product of the process, it does not detail the specific characteristics that the artefact should have.

With the intention of filling this gap we utilize the communities of practice theory as a framework to understand and make explicit the knowledge processes that lie camouflaged into ISD processes and artefacts. The main contribution of the article is to make explicit the intertwined knowledge nature of ISD process and product and the reasons and the conditions under which it is convenient to consider knowledge creation between the communities of developers and users as one of the main goals of ISD.

The proposed perspective emerges from a theoretical deductive reasoning. According to an interpretative viewpoint, an ISD project is then analyzed to illustrate the practical applicability of the perspective. The empirical evidences show some preliminary suggestions for the planning of the ISD activities.

Keywords: information system development, knowledge, communities of practice, boundary objects

1. INTRODUCTION

The development of proprietary information systems has been and continues to be a very complex activity marked by few successes and clamorous failures (Mathiassen and Stage 1990, Standish Group Report 1995, Remenyi et al. 1997, Beck 1999, Fowler 2002). Despite a continuous search for better techniques, as Avison et al. (1995) point out, the problem is not in the tools used but in the lack of attention to organizational and individual issues and their interaction with technology. Responding to this situation, during the last two decades, research and practice have begun to consider the information system development (ISD) process from an interpretive rather than a positivist perspective (Gibson and Singer 1982, Winter et al. 1995, Mathiassen et al. 1991). A social constructivist approach to ISD, where systems and requirements emerge from the interaction of multiple stakeholders, has begun to be preferred to an objectivist approach that considers requirements as exogenous factors, existing outside the interaction of individuals. Hence, the attention has shifted on to user participation that has been recognized as either the panacea for system success (Avison et al 1995, Winter et al 1995) or as a problematic issue (Markus and Mark 1994, McKeen and Guimares 1997). This interaction between users and other different groups (e.g. developers, managers, analysts, etc.) can be considered as the minimal activity to bring about just enough knowledge exchange to make the system requirements emerge. Within this view it has become very important to address the issue of how to facilitate the process of knowledge creation and exchange among the stakeholders involved in ISD projects (Bolici & Carugati, 2006). Knowledge issues in ISD have been treated both explicitly (e.g. Mathiassen et al. 1991, Beck 1999) and implicitly (Avison & Fitzgerald 1995, Avison & Taylor 1997). This is because the successful development of information systems depends both on the assimilation and combination of knowledge coming from different domains and on the intangible, cognitive and social nature of some ISD goals.

Interestingly the literature about system development methodologies is quite sparse in talking exactly about the system: the product and goal of the development process. Even if at the offset the research carried out in the field of information systems development focuses on the creation of the IT artefact, in reality it is said very little about the artefact itself (Orlikowski and Iacono 2001). Even the research work carried out on prototypes, incomplete IT artefacts that can be used as demonstrative tools of the behaviour of part of the system (Mathiassen and Stage 1990), does not specify how this artefact-prototype has to be but rather the process within which its use can be embedded. The taletail sings of this situation often are sentences of the type: 'prototypes can be used to …" (e.g. Mathiassen and Stage 1990, Boehm 1988, Davis 1982, Floyd 1987, Gladden 1982, Brooks 1987) and are in stark contrast with lack of sentences of the type: "according to this context prototypes should be …". In effect having reviewed an excess of 35 papers on ISD methodologies ranging from those most quoted among researchers (eg. Boehm 1988, Avison et al. 1995) to those most quoted by practitioners (e.g. Beck 1999, Fowler 2002) we have failed to find any description of how the software or the prototype should be (which specific characteristics it should have).

A similar single-facet approach exists in the literature about prototypes, which for historical reasons transcends ISD and it is most rich in the industrial design literature. Literature that describes objects to exchange knowledge appears disconnected from the processes that generate them (e.g. Star 1989, Winograd and Flores 1986, Boland and Tenkasi 1995, Carlile 2002). This literature highlights the existence of different ways of perceiving reality and presents objects as a way to bridge these views but it does not provides us with a path to follow for their construction. If the first set of literature is process-centric then this latter is overly object-centric.

Despite the existence of this dichotomy, recent research (Wenger 2000, Brown & Duguid 2001, Carlile 2004) presents knowledge not only as the result of a process of accumulation but also as localized and embedded into practice and into the objects that characterise it. According to these studies history, practice, knowledge, and objects are inseparable. A serious approach to ISD has therefore to take into account both the creation process and the object created in order to support the sought-after constructivist approach to ISD presented above.

The goal of this article is therefore to bridge the dichotomy process-object understanding and making explicit the knowledge processes that lie camouflaged into ISD processes and artefacts. The main contribution of the article is to make explicit the intertwined knowledge nature of ISD process and product and show that with this view it is convenient to set "knowledge creation" as one of the goals of ISD efforts. Providing a critical view on current ISD mainstreams, we would present a framework in which all the keys elements in a system development activity are taken in consideration. It is our belief that by serving the need for a constructivist approach to system development providing an integrated view of process and object we can present elements to improve the chances of success of ISD projects. In order to achieve this goal we draw on the well established body of knowledge of communities of practice (CoP) to provide us the framework to understand the ISD phenomenon from a knowledge perspective.

The article presents a theoretical discussion and uses a case as an example of application. The article is structured in the following way: we first present the theory of communities of practice showing that this framework is fit for making sense of ISD encompassing process and product. In section 3, we adapt the framework to ISD presenting a process knowledge creation based on reification and participation. In section 4, we present a case study with the scope of illustrating how the framework can be used in practice. Remarks and limitations of the application of the knowledge perspective in ISD are provided at the end.

2. THEORETICAL FRAMEWORK: COMMUNITIES OF PRACTICE

The ISD process is carried out by the interacting work at least of two, broadly defined, groups: the users or customers and the developers. The main reason why knowledge exchange is important throughout the whole ISD process is that very often it is difficult for the users to define the requirements of an IS at the beginning of a project. This is because when dealing with a new situation, like it is often the case in projects of software development, the possibilities granted by the technology are not clear to the users until they can see the system in operation. At the same time developers often do not have sufficient knowledge of the application domain and knowledge must be gained from the users in order to clearly understand goals and objectives for the IS. Some form of interaction between the development group and the user group is therefore required where the final characteristics of the IS emerge from mutual learning (Boland, 1979). However knowledge exchange between groups is very difficult to achieve. Groups that work together over periods of time begin to develop their own common language, methods, values and beliefs and all together these elements form the groups' worldview. According to this view, knowledge does not exist as an objective factor but only in the act of participation in complex social learning systems where knowledge emerges from the interplay between social competences created over time and the ongoing personal experiences of the people in the group (Richardson & Courtney 2004; Bolici, 2005). These groups, known as communities of practice (Wenger, 1998), have their own specific identity which is highlighted and preserved by social boundaries created to distinguish themselves from other groups.

Within these boundaries different communities of practice develop different kinds of knowledge which in turn contributes to thicken the boundaries even further. These specific knowledge boundaries are a source of separation, disconnection, and misunderstanding but they can become also areas of unusual learning, places where new possibilities might arise (Richardson & Courtney 2004, p. 233). Therefore boundaries also connect communities (Wenger, 1998; Carlile, 2002) and offer learning opportunities valuable in their own right. It is on these boundaries that the knowledge exchange of interest in ISD takes place.

2.1 Crossing Knowledge Boundaries in Communities of Practice

Wenger (1998) has pointed out the importance of the social processes of knowledge and interpretation management within and between the community of practices (CoP). He has introduced the concept of "negotiation of meaning", as a continuous and dynamic process of mutual agreement among different persons or CoPs. This concept of *meaning negotiation* has a strong connection with our analyses, since --as we have already presented it-- the ISD process could be considered as an activity in which at least two different CoP (developers and users) are involved. In order to facilitate the exchange of information and knowledge between user's and developer's CoPs is important to create some kind of connections between their groups. Crossing the boundaries between CoPs the ISD process could benefit from the CoPs' variety and differences of experiences and knowledge. Moreover, ensuring an effective negotiation of meaning process means to have higher chance to reach a common and shared understanding of all the activities in an ISD project, starting from the system goals definition.

As Wenger (1998, 2000, et al 2002) suggested, the negotiation of meaning could be represented by the interaction of two processes: participation and reification.

"*Participation* refers to a process of taking part and also to the relations with others that reflect this process. It suggests both action and connection." (Wenger 1998, p.55). Participation is an active process through which the members of different communities become part of each other trough mutual recognition and interaction. Participation should be viewed as an active process in which the possibility to mutually recognize, identify and accept members of different communities is the characteristic element.

The concept *reification* is the mental process by which it becomes possible to translate in real terms something that only has an abstract existence. "We project our meanings into the world and then we perceive them as existing in the world, as having a reality of their own."(Wenger 1998, p.58). Practically this process consists in molding our experiences through the production of objects that will freeze our own experience.

Both participation and reification contribute to connect different communities through their boundaries. Wenger (1998) considers two different mechanisms of connections among communities: brokering (based on the participation process) and boundary objects (based on the reification process).

Brokering is the possibility to belong to different communities and to be able to transfer different practices from one to another. Brokering process consists in translating, coordinating and aligning different perspectives from multiple communities. Hence, the persons that bridge different CoPs trough their practices --the brokers-- could create new connections among communities, facilitating the coordination and discovering new shared perspective. As participative activities, the brokering processes are very complex task to be accomplished since the broker needs a social legitimisation to bridge the communities. The different communities should trust in the not opportunistic behaviour of the broker and in its aim to provide better solutions for all the different communities involved in the negotiation of meaning process. Hence, the brokering activities is very dependent on the relational skills of the broker and his ability to motivate for cooperation.

Boundary object are those objects that allow the coordination of the perspectives of different communities (Star 1989). Inter-group connections created in this way are "reificative" since they embody in objects (real or reified) ideas and concepts that can be shared by groups that do not normally use the shared practices. Boundary objects enable conversation by presenting a reified representation of practices without enforcing a unique interpretation of meanings. This is especially necessary when doing ISD, since it is desirable that developers and users, while learning from each other, still maintain their own separate understanding of their own practices. Star (1989) and Wenger (1998) indicate four qualities for objects to work as boundary bridges. First *modularity*: different perspectives could be combined inside the object in a modular way (each perspective could add an element to the common object). Second *abstraction:* only some --and not all-- the characteristics of the different experiences are represented in the object. Real differences are needed to make the object interesting for a specific purpose, but a common ground is needed because otherwise the object will not be understood at all by one of the parties. Third *adaptation*: boundary objects could be used in a variety of different activities. Fourth *standardization*: the intangible resources in the object are formalized in a standardized structure, so each group knows how to use the object in its own context.

Participation with its supporting brokering activity and reification with its supporting boundary objects facilitate different kinds of knowledge exchange and require different focus. For instance, reification creates links among communities which go beyond the limitations of time and space typical of participative connections. On the other hand, reification (or boundary object) could be easily misunderstood if it becomes an isolate object without any kind of individual participatory engagement. Consequently, participation and reification are both distinct and complementary. Ideally, in order to have effective knowledge exchange they cannot be considered in isolation: they come as a pair. To understand one it is necessary to understand the other one. To enable one it is necessary to enable the other one. They come about through each other, but they cannot replace each other. It is through their various combinations that they give rise to a variety of experiences of meaning. Figure 1 below shows graphically the components of meaning negotiation in terms of participation, reification and their components brokering and boundary objects.

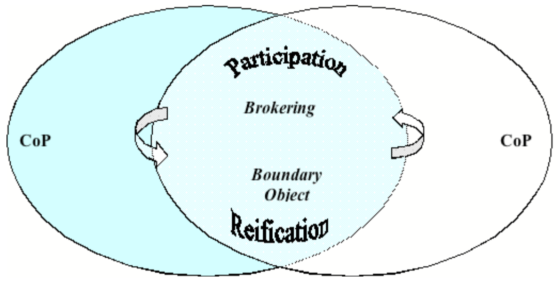

Figure 1. Meaning Negotiation with enabling mechanisms

The theory of communities of practice, focusing on the meaning negotiation through participation and reification, provides therefore a complete framework to make sense of ISD according to the social constructivist paradigm. At the highest level, meaning negotiation is what is required to make a proprietary information system emerge from the practices of users and developers. At the lower level, participation can be mapped directly to the participation required in ISD projects and reification can be mapped to system creation. The framework of communities of practices, seeing both the activity and the product as antecedents and enablers of knowledge exchange, provides us with a complete lens to bridge the gap between the process and the product that we highlighted in the introduction in the dichotomy methodology-system.

3. KNOWLEDGE AS GOAL IN ISD

According to the ISD methodology used, the ISD process can vary radically. However, there are four basic activities that are shared by structured and agile methodologies:

- The groups of developers and users meet and discuss the requirements for the information system.
- The developers discuss among themselves how to address the requirements.
- Each developer develops the part of the information system of his/her competence.
- The developers and the users meet to test the system and eventually implement it.

If a pure waterfall model is used the four steps will be performed once and if the spiral model or prototyping are used the four activities will be performed multiple times. If agile methodologies are used, e.g. XP, the four activities will present overlaps and become very blurred but are still distinguishable if observed with a very fine lens.

These activities can be reinterpreted through the framework of communities of practice presented above. In particular it is possible to apply the concept of participation and reification to view the process of creation of an information system. This can allow us to see the ISD process as a process of negotiation of meanings and therefore allow us to view the purpose of ISD as knowledge creation instead of artefact creation.

The idea of interaction between users and developers is well accepted in the ISD literature as a condition to system success but the CoP framework shows that the traditional users-developers relation is only a limited view of what participation can be. Participation is viewed as a complex and on-going phenomenon; participative activities are realized at different organizational levels: the user-developer relationship but also among users and among developers as separate communities. As Wenger (1998, p.56) defines, the factor that characterizes participation is the possibility of *mutual recognition*. "When we engage in a conversation, we somehow recognize in each other something of ourselves, which we address" (Wenger, 1998, p.56). In this way we become part of each other and participation could be considered as source of identity. Hence, considering participation activity during an ISD project consists in planning the involvement of some brokers across the different communities. Participating in different social communities, brokers shapes their own experiences (e.g. gaining a different perspective) and they also shape those communities.

If the idea of boundary object is applied to ISD, literature can be used to highlight some specific characteristics that the artefacts used in ISD should have (adding to the four qualities detailed above from Wenger (1998) and Star (1989)). Firstly, a boundary object must be *visual* (Brooks 1987, Carlile 2002). Visual artefacts are easy to inspect and quick to understand. Secondly, a boundary object must be *usable/functional* (Brown & Duguid 2001). This characteristic responds to the need of exchanging knowledge embedded in practice. The users must be able to use them in order to enact their daily routines and understand the equivoques of the developers and imagine the new possibilities that the software might provide. Finally, a boundary object must be *up-to-date* (Carlile 2002). If this is not the case the developers will be focused on newer problems and will miss the importance of the users' feedback. Furthermore, if the users evidence some flaws in the prototype the developers might get defensive because they will feel that the work and knowledge they invested is lost (Carlile 2002). The use of an up-to-date prototype will help to mitigate this form of resistance. The level to which an object fulfil these characteristics allows to "rank" objects according to their potential in facilitating knowledge exchange during ISD. Written documents have very limited effect. Diagrams and models are more efficient providing a visual representation but are not functional. Software prototypes can allow to exchange knowledge at the pragmatic level provided that they are visual, usable, and up-to-date. As Lanzara and Morner (2003) showed, artefacts could be "dynamic vehicles" for knowledge creation and sharing: "Artefacts play the role of media though which humans communicate and act" (p.3).

Artefacts then are boundary objects, and designing them is design for participation. Using a boundary object as part of CoP's practices enables the negotiation of meaning process between different communities and facilitates the activation of the participation process. Hence, the crucial issue in ISD activity is the relationship between the practices of design and the practice of use. Connecting the communities involved, understanding practices, and managing boundaries becomes a fundamental a design task. It is then imperative to consider a broader range of connections beyond the artifact itself, both to reconcile various perspectives in the nexus and to take advantage of their diversity (Wenger, 1998).

The CoP framework can be used to reformulate the four ISD activities (see beginning of this section). The discussion of the requirement involves both the user and the developer CoP. It involves participation in the traditional sense (co-location) and brokering is facilitated by the presence of both groups. It involves also reification as ideas are normally sketched on boards and participants begin to create their expectations. The second activity involves participation in the developers CoP and reification in terms of creation of models and blueprints of the coming system. Brokering can be used if a broker is present. The third activity is typically a reification activity where developers create the code and transform ideas into software. Concurrently the users participate in their CoP and experience reification as their expectations for the software concretize. The presentation and testing is an activity where again we have participation and reification with both CoPs present. In this phase the software can be used as a boundary object and the users must be given the possibility to use the boundary objects to understand the implications for their practice and to be able to criticize them convincingly.

The CoP framework can also be helpful to guide decisions in terms of the shape (linear vs. cyclical) and the timing (slow vs. long) of the ISD process. It takes time and multiple encounters for one group to understand the others' practices and accept their knowledge as valid (Boland & Tenkasi 1995). Therefore the process should be cyclical and include multiple iterations. Furthermore prototypes should be shown when the improvements (respect to the previous time) are not dramatic, so that the users can relate to the changes and give constructive comments. Proponents of agile methodologies suggest to present software prototypes every three to four weeks (Beck 1999, Fowler 2002).

ISD with participation and reification of the CoP framework is shown in figure 2 (derived by Carugati, 2004).

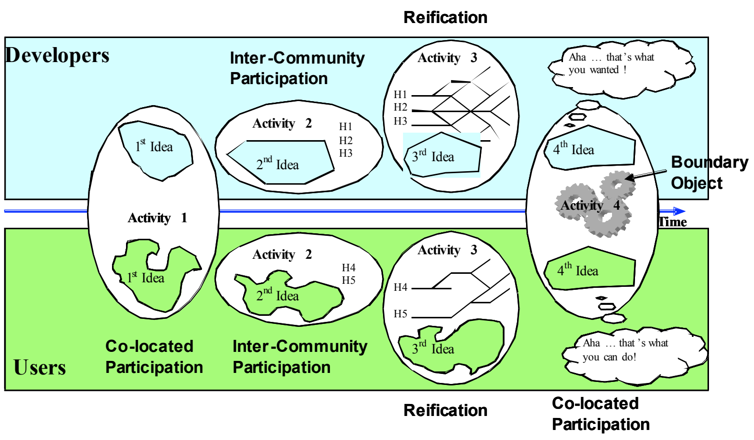

Figure 2. Participation and Reification in ISD (based on Carugati, 2004)

Using the community of practice framework shows that ISD projects are characterised by a continuous evolution in the knowledge of users and developers but that this evolution has to be channelled towards a constructive goal. The second and third ISD activities are not simply the execution of designs but complex processes of intercommunity participation and reification: for the developers reification is in the creation of the actual artefact and for the users reification happens when they create their own expectations. The reification process contributes to the creation of IT artefacts by the developers that can be very different from what is expected by the users. If the process of knowledge creation is not well understood and guided then there is a risk that it might become dysfunctional and the meaning negotiation will not take place. The following section presents such a case where the CoP framework is applied to highlight dysfunctional points.

4. EMPIRICAL ANALIYS

This article is intended to present a theoretical discussion nonetheless it seems appropriate to present how the CoP framework can be used to highlight the knowledge dimension of a real ISD project.

4.1 The Shipyard Project: Application of the CoP Framework

The example comes from a project where one of the authors was participant observer from 2000 to 2003. The project goal was to develop software for scheduling and control of the row material inventory of a shipyard. The project presented two characteristics that made it a good example for the arguments presented in this paper. The first point was that the IS under development was based on a very advanced mathematical technique, whose properties in terms of needs and performances were new to the users. The second characteristic was that the development team was external to the shipyard, and the developers were working in different locations. Furthermore the developers were university researchers (four universities with eight people) with background in system development. These characteristics point out that the problem of knowledge creation and exchange was not only relevant but key to the success of this project. The first characteristic (novelty) made it important for the users to learn about the mathematical technique in order to be able to specify the requirements; the second (delocalization) made it important for the developers to learn from each other about their work.

In the following we are going to present the project seen through the communities of practice lens to provide an immediate insight on how the framework can be used. To achieve this, we highlight the communities involved and discuss the participation process with the brokers and the reification process with the boundary objects. Finally we present the results of the project in terms of meaning negotiation.

Communities of Practice. The two communities that emerged during the project were the shipyard employees and the developers. The employees had an average tenure in the company of 15 years and had been working in material handling for at least 5 years. The developers had an academic background and were either senior researchers or doctoral students. Within each group the members shared practices, problem solving approach, view of the world and both groups had a "we versus them" attitude (boundary). At the beginning of the project there are two different communities: the developers from the academic institutions and the organization's employees. At this stage, these communities identify themselves on the base of their background, experiences and work-place. The employees had an average tenure in the company of 15 years and had been working in material handling for at least 5 years. The developers had an academic background and were either senior researchers or doctoral students. As the process moves from the conceptualization phase to the coding phase, the sharing of the practices among all the participants reshapes the communities' borders.

Participation. The project was carried out in a traditional waterfall fashion (with the different phases of requirement definition, system design, system development, and testing). Inter-community participation was very sparse throughout the all project. The pinnacle of participation was when the developers were dispatched for one week to the users' stations to observe their work at the beginning of the project. This remained the only contact throughout the project between developers and the complex working environment of the users. The requirement definition was carried out by the developers alone and when asked to participate to the requirements discussion the shipyard project manager (pm) replied:

> *I will not come to the meeting on Tuesday. After all the idea with the meeting is that the research partners (the development group) will prepare an action plan.*

Participation among the CoP of developers changed nature along the project. In the requirements collection activity it was limited to few meetings (on average 1 every 3 weeks for the first 6 months); then, during the design activity, it was quite intensive but only using emails; during the coding phase there was a shift to very intensive work carried out co-located. This last phase was punctuated by prototypes presentations. Prototypes became occasions for participation only quite late in the project: in the 16th, 19th and 21st month. However participation in these occasions did not result in knowledge exchange as the presentations were monologues from the developers to the users: no debates emerged nor practical tests were carried out by the users.

Broker and Brokering. The one figure that acted as broker in the project was the developers' pm. The pm had developed software for the shipyard before and had been employee there for three years. At the time of the project he was employed in a consulting company. The pm was living 2 km from the shipyard and the proximity and the acquaintances he had granted him very easy access to the company. The pm had been an academic and held a Ph.D. in computer science. He knew well some of the people in the developers team since before this particular project. Having knowledge of both the users' and the developers' world, the pm was potentially a well suited broker for this project but he did not always played this role. During the requirement definition and the design activity he was a very active broker but as the software began to be coded and the project encountered problems he began to act less and less as a bridge and left the developers autonomy concentrating on the shipyard. Without broker the developers advanced at higher speed but towards their vision of the goal something that would result very different from the users' expectations. As pm he led the prototype presentation meetings and these were good occasions for brokerage but since he limited the brokerage to these occasions the developers did not pick up on the users' remarks later in the coding. The intermittent brokerage activity gave the impression that it facilitates knowledge exchange but in reality it hindered the process. The problem with brokerage and lack of knowledge exchange was highlighted in an interview with a developer towards the end of the project:

> *I think that the [problem definition] could have been done better if someone that knew about the problem could have interacted with us from the very beginning. Instead we went around without really knowing what we were looking for. The problem was that the shipyard did not know what we needed and we did not know either.*

Reification. After the design activity the developers, singly or in couples, created the code that would become the software. The transformation of ideas into code was a process of reification. The first demonstration was continuously delayed to improve the software but the more the software was improved the more complex it became for the users to understand problems and opportunities. During the development project the development team held three demonstrations of the software (see above) with the goal of collecting feedback from the users. These participative activities were followed by reifying activities of code creation. However when comparing the scripts of the meeting or the following emails of the users with the changes in the code no appreciable changes can be noticed rather the code continued to reflect the original plan of the developers. This shows that the reification process was active but not across groups.

Boundary objects[1]. During the project it was soon realized that the advanced mathematical technique used required to be demonstrated through software prototypes. In this sense the idea of using reification for knowledge exchange was clear to the project participants but the execution failed to reach the target. The software that was presented to the users was visual but it was too complex for the users to use it. The testing phases were organized like presentations where the developers used the software and presented the results in form of numerical tables. The software was up-to-date but it contained simplifications with respect to the users' practices and changes to these simplifications were extremely difficult to implement. The developers were therefore very defensive towards making changes to these simplifications. Finally the software had many features that worked as black boxes, for the users it was very difficult to know how the results where generated and since they could not understand and could not use the software they did not have a possibility to either learn or teach. The software, even if it was called a prototype, was not conceived nor used as a boundary object: being very complex and presented in the wrong fashion it increased the knowledge boundary rather then bridge it.

Meaning negotiation. Finally the software was never implemented. The negotiation of meanings – so vital in a project with a complex content and a complex organization – never took place. The components of the meanings negotiation were missing: participation was very scarce and brokering intermittent; reification was asymmetric and the boundary objects reinforced the boundary instead of destroying it. Approaching the end of the project, the shipyard project manager was interviewed and asked what he had learned about the use of the mathematical technology. The reply was that his knowledge was still superficial (after a 3 years project). This answer shows that failure was not only in bringing the software to implementation but also in creating the knowledge to approach similar problems in the future. In conclusion to achieve negotiation of meanings in such complex projects it is necessary to have both participation and negotiation and to have them continuously. Brokering cannot be intermittent and software prototypes must have the characteristics of boundary objects and be embedded in a process that allows knowledge barriers to fall. Failure to achieve any of these can lead to failure in shipyard case and will probably lead to failure in any other case.

The community of practice framework could be a very effective tool to highlight different means of knowledge exchange and the process in which these means are used. For our proposition – that in ISD it is convenient to aim at knowledge creation and exchange – the framework provides an interesting lens for the qualitative appreciation of this performance.

4.2 Discussion

Before passing at the conclusions we want to present some of the considerations that we have made regarding two topics of this paper: the first about the positive and negative aspects of using knowledge as a goal in ISD; the second about the differences and similarities of boundary objects and prototypes.

Positive and negative aspects of the knowledge goal

The proposed perspective and model of ISD, based on knowledge creation process, assists in obtaining a better representation of the design activities. This point is still more important since the ISD projects have a very low success ratio, and the debate on the different possible goals that an organization should pursue is still open.

Considering knowledge creation as an important ISD goal, the organization will push the different actors involved in the design activities to interact and to cooperate in a more intensive and dynamic process. Since cognitive assets are strongly dependent on the variety of interacting actors, an effective communication and a valuable social relation between different communities could be a key factor for succeeding. Moreover, as we presented above, the negotiation of meaning --with the participation and brokering activities-- is a very complex process that overtakes the traditional perspective of a passive presentation of the (on-going) product to the users. Instead negotiation of meaning consists in an effective and deeply interaction and combination of the practices between communities. Crossing CoPs' boundaries becomes a possible source of knowledge creation and exchange. Another advantage of this perspective is that both the developer and user community have to force themselves to be clear in their own explanations, requests and capabilities.

[1] As a reminder, an artifact to function as a boundary object has to be: visual, usable, and up-to-date.

Infact, an effective and valuable connection between practices could be reached only through a fair cooperation where opportunistic behaviours are discouraged. The different communities involved in the project are force to negotiate in a dynamic and interactive way the meaning of their own practices and so of the ISD project. Moreover an interesting aspect coming from this perspective is that also when the ISD project is not a complete success or is a failure (maybe the system is not completely implemented, or not all the functions are ready, the product is not interesting for the user, etc.) the organization could earn something. The companies could learn expertise in managing knowledge factors in specific context, in order to avoid committing the same mistakes next time.

The proposed perspective has also different limitations and negative aspects. First of all, this ISD process is time-consuming. The continue interaction, the dynamic negotiation process, the possibility continuously change the direction of the developing activities could significantly influence the time needed for developing the system. Moreover, a participative process needs a lot of management resource in order to effectively organize all the cooperating activities and to drive the efforts of all the actors towards a common and valuable aim. Furthermore, knowledge --as all the intangible and cognitive factors-- is quite difficult to measure. In such a way it becomes hard to give objective and quantitative goal for the project and the management activities turn into very complex and subjective activities.

Prototypes and Boundary Objects: Can a Name Change Anything?

A point might be raised that boundary objects and prototypes are – talking about software – the same thing and that there is no need to make further distinctions. We believe that there are indeed overlaps but there are also differences that need to be clearly understood if the knowledge goal in ISD has to be effectively pursued.

A prototype can be defined in different ways according to the purpose. It is normally defined as an incomplete, test-version of a non-finished product made to illustrate the behaviour of a part of the system (Mathiassen and Stage 1990, p. 483) or made to test the principles of a solution before the final product is constructed. In this case the prototype responds to questions of feasibility. From a managerial point of view instead, a prototype can be used to test the effectiveness or efficacy of a solution in solving a specific problem. In the definition of requirements a prototype can be used to answer questions like: Is the problem perceived correctly? Does it solve the right problem? Are the specifications correct? For testing purposes a prototype can be used to test feasibility and user acceptance. For all these purposes, prototypes are seen as means to create knowledge.

It seems that a prototype and a boundary object can solve the same problems. In reality the case presented in section 4 is a concrete example that even if software is called a prototype and answers questions of feasibility and performances it can still hinder strongly any form of knowledge exchange. While prototypes have a very positive connotation – use them and knowledge will be exchanged! – boundary objects highlight the dark side of the problem: namely the existence and persistence of boundaries. Prototypes can help answer questions about feasibility and effectiveness but if they lack the characteristics of being visual, usable, or up-to-date they can perform very poorly as knowledge bridges. In conclusion while certain prototypes have the characteristic of boundary objects not all of them are so.

Furthermore the name itself might make a difference because "boundary object" focuses the question of "are we exchanging knowledge?" which is specific of both process and product rather than of "does it work?" or "how good does it work?" which are specific of the product alone.

5. SUMMARY AND CONCLUSIONS

In the development of software, knowledge exchange proves to be a critical element to accomplish the project successfully. In the paper we have proposed and adapted the communities of practices as guiding framework to analyse ISD cases and monitor where and when knowledge is created. CoP provides a good framework to make sense of the ISD projects as it provides an integrated view of process and product that we found missing in traditional IS literature.

CoP highlights the existence of two processes, participation and reification, that are key in the negotiation of meanings between different groups. Participation is seen as an active process through which the members of different communities become part of each other trough mutual recognition and interaction.

Participation in CoP is therefore similar to the traditional concept of participation in ISD but more specific to knowledge exchange. With a vision of knowledge as something inseparable from practice, boundary objects as reificative means have been presented as particular instances of software prototypes that bridge the knowledge boundaries in the practices of developers and users.

The adaptation of the CoP framework to ISD allows to better guide, manage, and monitor ISD projects because it assures the support to the social constructivist approach for system creation. One example has been used to complement the theoretical discussion where it has been shown how the CoP framework can be used in practice to make sense of and evaluate ISD projects. It is shown that the separation of process from product is not a just theoretical hole but their integrated consideration augments our predictive and evaluative capabilities and it is therefore a key element in pursuing the (larger) system success.

References

Avison D.E. e Fitzgerald G. (1995), Information System Development. Methodologies, Techniques and Tools, II ed., New York, McGraw-Hill.

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

Avison D.E., Wood-Harper A.T.; Vidgen R.T.; Wood J.R.G., 1998, A further exploration into information systems development: the evolution of Multiview2, Information Technology & People, Vol. 11 No. 2.

Beck K., 1999, Extreme Programming Explained: Embrace change, 1.ed., Addison Wesley.

Boehm B., 1988, A Spiral Model of Software Development and Enhancement, IEEE Computer.

Boland R. J., Tenkasi R.V., 1995, Perspective Making and Perspective Taking in Communities of Knowing, Organization Science, Vol. 7, No. 4.

Boland R.J. 1979, "Control, causality and information system requirements", Accounting, Organizations and Society, 4: 259-275.

Bolici, F, 2005, Organizational Knowledge Processes: Technology Role from Social-Practice Perspective, Proceedings of the 21st EGOS Colloquium, Berlin.

Bolici, F., A. Carugati, 2006, Combining Processes and Objects in ISD Projects: Aiming For Knowledge Creation, International Conference "Coordination and Cooperation Across Organisational Boundaries", Milan, 20-21 April.

Brooks, F.P, 1987, No silver bullet: essence and accidents of software engineering, IEEE Computer, Vol. 20, No. 4.

Brown J.S., Duguid P., 2001, Knowledge and Organization: A Social-Practice Perspective, Organization Science (12,2)

Carlile P.R., 2002, A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development, Organization Science, July/August

Carlile P.R., 2004, Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries, *Organization Science,* vol.15, n.5, p.555-568.

Carugati A., 2004, New Challenges for the Management of the Development of Information Systems Based on Advanced Mathematical Models, Unpublished Doctoral Dissertation, Technical University of Denmark, Lyngby, Denmark.

Checkland P., Holwell S., 1998, Information, Systems and Information Systems, Wiley, Chichester.

Churchman C.W. (1971). The Design of Inquiring Systems, Basic Books, New York.

Fowler M., 2002, The New Methodology, www.martinfowler.com (visited in June 2003)

Gibson C.F., Singer C.J., 1982, New Risks For MIS Managers, Computerworld "In Depth" section, April 1982.

Hughes J., Wood-Harper T., 2000, An Empirical Model of the Information Systems Development Process: a case study of an automotive manufacturer, Blackwell Publishers Ltd. Oxford.

Lanzara, G. F. & Morner, M. (2003). The Knowledge Ecology of Open-Source Software Projects. Proceedings of the 19th EGOS Colloquium, Copenhagen.

Lee, G. K. & Cole, R. E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case the Linux Kernel Development. Organization science : a journal of the Institute of Management Sciences, Vol.14, 633-649.

Mathiassen L., Munk-Madsen A., Nielsen P.A., Stage J., (1991), Soft Systems in Software Design, in: System Thinking in Europe, M.C. Jackson et al. (Eds.), Plenum Press.

Mathiassen L., Stage J., Complexity and Uncertainty in Software Design, in: Proceedings of the COMPEURO 90 Conference held in Tel Aviv, Israel, May 7-9, 1990

Orlikowski W.J., Baroudi J.J., 1991, Studying Information Technology in Organizations: Research Approaches and Assumptions, Information Systems Research, Vol. 2, No. 1.

Remenyi D., White T., Sherwood-Smith M., 1997, Information Systems Management: The Need for a Post-Modern Approach, International Journal of Information Management, Vol. 17, No. 6.

Richardson, S. M., Courtney J. F. (2004) A Churchmanian Theory of Knowledge Management System Design, 37th Hawaii International Conference on System Sciences, Hawaii.

Standish Group, 1995, The Standish Group Report, http://www.scs.carleton.ca/~beau/PM/Standish-Report.html (01.03.2004).

Star S.L. e Griesemer J. 1989, "Institutional ecology, translation and boundary objects: amateurs and professionals in Berkeley's museum of vertebrate zoology", *Social Studies of Science*, 19: 387-420.

Star, S.L., 1989. The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving, in Huhn,M., L.Gasser (eds.), Readings in Distributed Artificial Intelligence. Menlo Park, USA: Morgan Kaufman.

Weick K.E., 2001, *Making Sense of the Organization*, Blackwell, Oxford.

Wenger E. 1998, Communities of Practice. Learning, Meaning, and Identity, Cambridge University Press, NY.

Wenger E., 2000, Communities of Practice and Social Learning Systems, Organization, Vol. 7, No. 2.

Wenger, E., McDermott R. e Snyder W.M. 2002, Creating Communities of Practices, H.B.S. Press, Boston.

Winter M.C., Brown D.H., Checkland P.B., 1995, *A Role for Soft Systems Methodology in Information Systems Development*, European Journal of Information Systems, Vol. 4.

The authors wrote the whole paper in complete collaboration. In particular sections 1, 4 and 5 were written by Andrea Carugati, while Francesco Bolici focused his attention on sections 2 and 3.