Università Ca' Foscari – Venezia

L'ORGANIZZAZIONE FA LA DIFFERENZA?

IX Workshop dei Docenti e dei Ricercatori di Organizzazione Aziendale

7 – 8 Febbraio 2008

Track: Modelli organizzativi per l'innovazione e per il trasferimento tecnologico

KNOWLEDGE CREATION AS AN ISD GOAL: LESSONS FROM AN EMPIRICAL STUDY

FRANCESCO BOLICI

Università degli Studi di Cassino

frrancesco.bolici@eco.unicas.it

ANDREA CARUGATI

University of Aarhus

INTRODUCTION

The development of proprietary information systems has been and continues to be a complex activity marked by few successes and clamorous failures (Mathiassen and Stage 1990, Standish Group Report 1995, Beck 1999, Fowler 2002). Interestingly the literature about system development processes is quite sparse in analyzing simultaneously the different aspects of an ISD process. Most of the contributions consider the characteristics of the product (software) as factors separated and isolated from the flowing and social nature of the development activities. Hence, on one side the contributions focused on the development activities pay scanty attention to the software characteristics and, on the other side, the research focused on the software artefact neglect the processes and the social interactions that leaded to that object. Considering the management-centric literature, even the research works carried out on prototyping do not specify how this artefact-prototype has to be but rather the process in which it is embedded. The taletail signs of this situation are sentences of the type: 'prototypes can be used to …" (e.g. Mathiassen and Stage 1990, Boehm 1988, Brooks 1987) and are in contrast with lack of sentences like: "according to this context prototypes should be …". Having reviewed an excess of 35 papers on ISD, focused on the development processes, 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 should be (which characteristics it should have). A similar single-facet approach exists in the literature about software (or code/algorithms). Literature that describes software appears disconnected from the management processes that generate it (e.g. Star 1989, Winograd and Flores 1986, Boland and Tenkasi 1995, Carlile 2002). If the first set of literature is management-centric then this latter is overly object-centric. However, in ISD projects, we find that there is a strong connection between how the activities are carried out and the final product that is developed (Lanzara & Morner, 2003). The distinction between organizational and technical factors is therefore more a human creation than a phenomenon observed in actual ISD projects. In other fields, not strongly connected with ISD, this gap has

been filled considering knowledge as localized and embedded into practice and into the objects that populate it (Wenger 2000, Brown & Duguid 2001, Carlile 2002). According to these studies management processes, practices, knowledge, and objects are inseparable.

The aim of this article is to bridge the dichotomy management-object in the ISD field. In order to achieve this goal we will analyse and make explicit the knowledge processes that link ISD artefacts and social processes. It is our belief that by serving the need for a constructivist approach to system development providing an integrated view of managerial processes and objects we can provide elements to improve the chances of success of ISD projects. In order to achieve this goal we draw on the body of knowledge of communities of practice (CoP) to provide us the framework to understand how objects and managerial practices are connected in ISD from a knowledge perspective.

THEORETICAL FRAMEWORK: COMMUNITIES OF PRACTICE AND ISD

The ISD process is carried out by the interacting work of (at least) two groups: the users and the developers. During ISD activities these groups must find a way to exchange their knowledge in order to achieve their goals (e.g. Avison et al 1995, Winter et al 1995). As Fitzgerald (1996) explains, the assumption that the developer can obtain detailed knowledge about the problem situation (Giddings, 1984) is questioned by Jones and Walsham (1992) who argue that there are limits to both "what can be known" and "what should be known". Moreover, Boland (1979) identifies the importance of how a situation is interpreted by the actors in the context, questioning the existence of a problem as an independent reality that can be autonomously modelled. Stage (1991) also argues that parts of the knowledge relevant to the organizational problem may be impossible to express[1] by descriptive

[1] For a deeper analyses of the dichotomy between tacit and explicit knowledge see Polyani 1958, 1983; Nonaka, 1994.

techniques. Knowledge exchange activities are important since the beginning of the ISD process, because the users usually face big difficulties in defining the requirements of an IS at the beginning of a project. This is because when dealing with a new situation 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 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 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 and Courtney 2004). 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 contribute 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 and Courtney 2004, p. 233). Therefore boundaries also connect communities (Wenger, 1998; Carlile, 2002) and offer learning opportunities. It is on these boundaries that the knowledge exchange of interest in ISD takes place.

Crossing Knowledge Boundaries in Communities of Practice

Following the considerations above, it becomes crucial to address the issue of how to facilitate the process of knowledge creation and exchange among the stakeholders involved in ISD projects. This is because the successful development of information systems depends on the combination of knowledge coming from different domains.

Wenger (1998) has pointed out the importance of the social processes of knowledge and interpretation management within and between the communities 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 ISD is 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 necessary to create some kind of connections between their groups. Crossing the boundaries between CoPs, the ISD process could benefit from the CoPs' variety of experiences and knowledge. Moreover, ensuring an effective process of negotiation of meaning leads to higher chances of reaching a common and shared understanding of the ISD activities.

As Wenger (1998, 2000, et al 2002) suggests, 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 connect through mutual recognition and interaction. 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 freeze our experiences.

Both participation and reification contribute to connect different communities through their boundaries. Wenger (1998) considers two different mechanisms of connections among communities: brokering (linked to participation) and boundary objects (linked to reification). *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. As participative activities, the brokering processes are very complex tasks to be accomplished since the broker needs social legitimisation to bridge the communities. Hence, the brokering activity 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 share practices. A boundary object must be *visual* (Brooks 1987, Carlile 2002) since visual artefacts are easy to inspect and quick to understand. It must be *usable/functional* (Brown & Duguid 2001) responding to the need of exchanging knowledge embedded in practice. and *up-to-date* (Carlile 2002) therefore embodying the latest knowledge produced. Boundary objects enable conversation by presenting a reified representation of practices without enforcing a unique interpretation of meanings. This is especially necessary when engaging in ISD, since it is desirable that developers and users, while learning from each other, still maintain their own separate understanding of their practices (Jones and Walsham, 1992).

The theory of COP, 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 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 needed in ISD projects and reification corresponds to system creation.

CASE STUDY

The case study concerns a project where one of the authors was participant observer from 2000 to 2003. The project goal was to develop software for planning and control of the raw material inventory of a shipyard located in Denmark.

The main characteristic of the case study was that the complexity of the planning and control task[2] required the use of a particular mathematical technique, known as combinatorial optimization. This technique -even though it could not provide the certainty of optimality- could provide a far better solution than any other heuristic solution available on the market. The novelty of the combinatorial optimization technique to solve planning problems was such that university researchers were required to carry out the development. The shipyard (SteelCo) was contributing with members from different functions: two projects sponsors, the inventory area manager, the inventory area supervisor, and the internal project manager from the planning department. An IT consulting company was contributing with the project manager. He had developed software for the shipyard before and held a Ph.D. in computer science. Among the team members, seven took active part in the development process. These were specifically: the project manager (Ted), the developer of the planning part of the software (Joshua), the developer of the control part (Walter), the person in charge with system modelling (Drew), the person in charge with the organizational change process (Tom) – henceforth we will refer to these five using the expression 'developers'– and the project sponsors for the client company (Rob and Jim). The developers never met before the project started. The project was planned to last three years.

[2] The problem is NP complete: the calculation of the exact solution would take infinite amount of time.

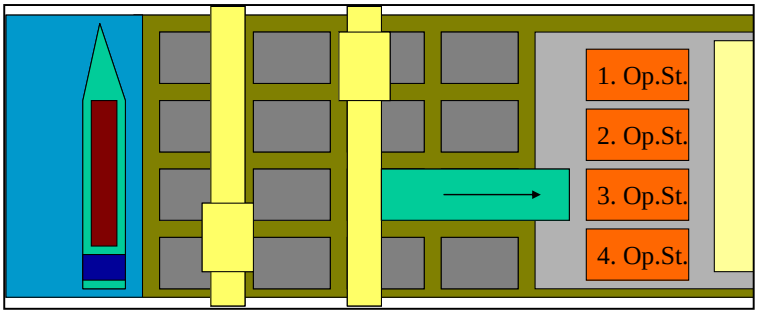

Figure 2. Inventory layout at SteelCo

Figure 2 shows the schema of the inventory layout. Starting from the left we have the sea with the ship that delivers the steel plates. The grey squares represent the 256 piles of steel plates organized like a chessboard (8*32). Each pile could have a maximum of 200 plates. The yellow bars crossing the inventory represent the two cranes that move the plates. The cranes cannot pass each other. The green horizontal bar represents the preparation line (prep-line) where the plates are cleaned and painted and prepared for the plasma cutters (Operation Station 1, 2, 3, and 4). The crane-operator does not know exactly in which pile a specific plate is located, he only knows that it can be found in about 4-6 different piles. It is left to the experience and memory of the crane operators to know and find where a plate is. Given the production needs of the plasma cutters, the job of the crane operators is to find the right plates in the piles moving the ones above to others piles. The main problem of the inventory is therefore the location of the plates and their movement so to facilitate the future work.

Research Methodology and Data Collection

Data collection during the SteelCo case was conducted throughout the entire duration of the project. During the project one of the authors performed activities of liaison between IS stakeholders on the client side and the development team. Data were gathered in form of direct observations, interviews, minutes of project meetings,

software versions and other documents. Observation took the form of participating in the project as a member of the academic team. During the collaboration, notes were taken of the work practices of both the SteelCo and developers groups. Twenty-seven semi-structured interviews of 60-80 minutes in length were conducted at the beginning and at the end of the project. All interviews were recorded and transcribed. Interviews were carried out with all participants and stakeholders both of SteelCo employees and of academic partners. Minutes were the standard documents produced for all the official meetings where all the participants were present. Moreover, we tracked the evolution of the code retaining every new software release (collected in the Concurrent Version System used by the programmers). As last source of information we gathered a variety of SteelCo internal documents as: work procedures, activity models, production plans, and specification for another inventory management software.

After the project was concluded the most active participants were convinced to provide all their e-mails for the development period. It was important to ask for the emails after the project end because this would avoid biases in the content, workarounds, or opportunistic selection. E-mails were an interesting data source because they contain information of direct communication among team members and therefore they help to support field notes and interviews' content.

We use qualitative techniques to analyse the data (Eisenhardt, 1989; Strauss and Corbin, 1990; Yin, 1994). The analysis was informed by our focus on the gap between managerial processes and objects in ISD. We first red all the field notes, the interviews transcripts, the meeting minutes, and the emails to identify trough the CoP lens the issues and topics related to the connection between managerial processes and objects. The coding was carried out once for every data source. The intention was to find overlapping or dis-proofing evidences across the data sources and ultimately present an accurate image of how process and product interact.

Case Presentation

The project started officially with a meeting held in January 2000 where all the parties involved were present. In this meeting it was decided to follow a development process that involved three steps: observation, modelling of processes and activities, and creation and presentation of software prototypes. The developers received from SteelCo a variety of documents to get started such production plans, standard operating procedures, etc. All this documentation turned out to be quite incomplete and outdated but gave an idea of how the company worked with ISD.

In February 2000 two developers (Ted and Drew) visited the inventory department of the SteelCo as observers to obtain a deeper knowledge of the company. Their aim was to understand the practices in action, the processes, and more in general the nature of the problem in order to collect the users' requirements. The following week Joshua, the programmer of the planning module, and Tom visited the company. During these visits the developers got acquainted with the environment and with the machinery and raw materials involved in the process. The developers interacted with all the people involved in the logistic activities from the crane-operators in the inventory to the workers on the prep-line. As the discussion with the workers happened while they were working, the operators explained the scope, the function, and the usage of different machineries.

In March/April the developers created the activity and process models of the inventory and presented them to the company's sponsor for feedbacks. The models were not enough to create a debate about the practices and were considered by the company a loss of time because they depicted known processes. Without debate and with unchanged models the developers had nothing else to proceed with than the models themselves. This situation created an unbalance in the developers group between the ones that wanted to start programming right away and those that argued against it because the context was not yet clear. In this debate the role of the project manager was essential because he was writing and emailing around the project plans

and therefore he tipped the equilibrium towards starting to program. The first official plan was sent out on April 5th 2000.

After the initial period the main goal became clear: make a software that could *plan* the movements of the cranes (provide a predetermined sequence of what to move, where, and when) and could *control* the execution of the plan. The control module was included to compensate for the variability of production processes (delays, breakdowns, missing plates) that would inevitably disrupt the plan during execution. The two parts of the software were conceived as two modules and assigned to the respective experts.

During a meeting held in May 2000 the discussion revolved around whether or not to include a visual simulator as an additional module. Sustainers of yes (easy to communicate, easy to validate) and sustainers of no (too complex, time consuming) fought lengthy on the issue until the developer of the control module took out his computer and showed a visual simulator of a robot that the shipyard was using. The visualization was familiar to the company people because it represented their robot and was familiar to the developers because one of them had programmed it. The visualization had such an impact that after there was complete agreement that the visual simulator had to be done.

The conceptualization phase ended abruptly in June 2000 with an email from the project manager to the developers. The email detailed a traditional approach to system development (waterfall) where developers had to work on the requirements and ideas collected so far and create the software. The summer 2000 is characterized by a low level of programming activity. Following the project plan, the project manager and Drew worked on the system architecture in order to assure that planning, control and simulation modules could work well together. The output of this activity was a document describing architecture and the common data model of the software. The components and their interactions are showed in figure 3.

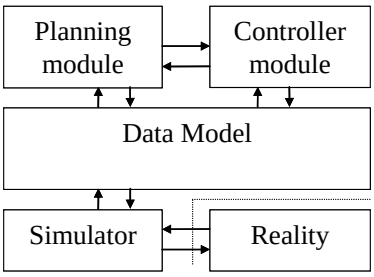

Figure 3. Architecture and data model

At the end of August a new project plan is sent out by the project manager to specify the new activities and their sequence. The project plan that covered the period from the 30th of August 2000 to the 16th of February 2001 had very strict deadlines. In this period the developers began to work concurrently on the code (Joshua and Walter) and on the use cases (Drew). However the scheduled plan was not respected. The period September 2000 – May 2001 is signed by the arrival of two more programmers responsible for the simulation module and one to help with the planning module. In this time there is no interaction with the users and only a very limited interaction with the sponsors. The programmers started working together using an online tool for version control (CVS). In this period –characterized by a very high level of email exchanges– the programmers began to isolate themselves from the others in the group. This email excerpt clarifies this idea:

It is clear now that it is going to be a hardcore hacker meeting on Monday where we will focus on coordinating the system development at all levels (Joshua, email, October 10th 2000).

As the developers entered the more complex coding phase, they began planning their activities at had hoc basis (coding, emailing, meeting, discussing) and stopped following the work-plan proposed by the pm. A mail from Walter explains this:

I am not advancing at all. All of my code must be re-written. I think we must drop the common data model and instead define our own interfaces. Now I have lost another day. Walter (12-10-2000)

Forced by the need to overcome certain difficulties[3], the programmers first moved towards an architecture where the planning and controller modules could work separately and only later they linked them with a "patchwork solution". This evolution is show in figure 4.

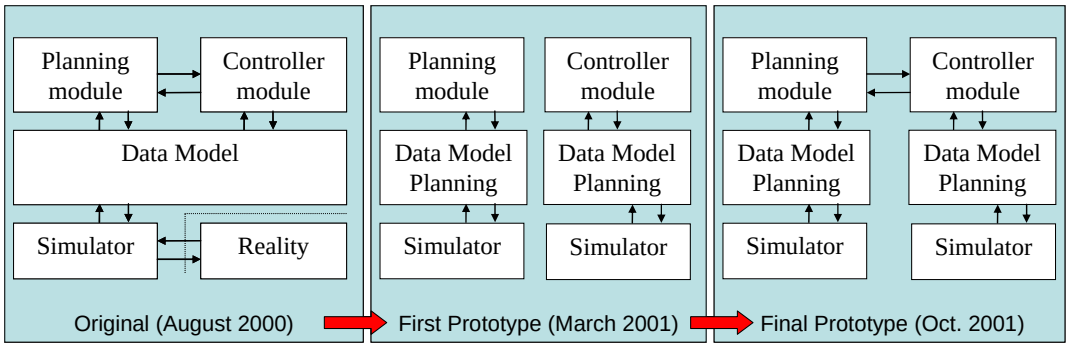

Figure 4. Evolution of the software architecture over time

The development team held three prototypes demonstrations in March, July, and October 2001. The idea behind the demonstrations was that the users could control the correctness of the software and could eventually specify new requirements. The three sessions functioned in a similar way. The testing was organized like a presentation where the developers showed pre-calculated runs to the users. The discussion was structured in a way that left to the users very little space for criticizing the validity of the simplifications in the software. One of the most important examples of this is represented by the company's requests for inclusions of disturbances (delays, machines breakdown, plates not found, etc.) in the simulator and the refusal to do so from the developers.

Finally the software was not implemented and when, approaching the end of the project, one author asked the shipyard's sponsors about the knowledge they acquired about the CO technique they replied that their knowledge was still superficial. This answer shows that not only did the project members fail to create the software but

[3] Among the many problems one was the length of the calculations. The developers had to ask and luckily obtained the chance to use a supercomputer to carry out all the tests in time.

they also failed to transfer the knowledge useful for approaching similar problems in the future.

EMPIRICAL ANALYSES

The Shipyard Project: Application of the CoP Framework

Communities of Practice. At the beginning of the project the data collected allow us to identify two different communities: the developers from the academic institutions and the SteelCo's employees. At this stage, these communities identify themselves on the base of their background, experiences and work-place. As the process moves ahead, the sharing of the practices among the participants reshapes the communities' borders. In the development team we observed the emergence of the "hackers" community.

Participation. The project was carried out in a traditional waterfall fashion. Inter-community participation was very sparse throughout the project. The pinnacle of participation was when the developers went for one week to the users stations to observe their work at the beginning of the project. Participation among the CoP of developers changed nature along the project. In the requirements collection activity it was limited to few meetings between the developers (on average 1 every 3 weeks for the first 6 months); then, during the design activity it intensified to a meeting every 2 weeks; during the coding phase there was a shift to very intensive work carried out through weekly meetings. This last phase was punctuated by prototypes presentations which became occasions for the entire project group to meet. However in these occasions participation was very limited as the presentations were monologues from the developers to the users: no debates emerged nor practical tests were carried out by the users. Hence these meetings could not be considered real participating activities since there was not real sharing of practices.

Broker and Brokering. The project manager was selected for his background characteristics having knowledge of both the users' and the developers' world. He

was therefore potentially a well suited broker for this project but our observations shows that 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 own vision of the project goal. The pm led the prototype presentations but, since he limited the brokerage to these occasions, this was not sufficient for achieving a real exchange of practices.

Reification. The transformation of ideas into code was the main reification process. The prototype demonstrations were 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. The demonstrations were planned as participative activities followed by reifying activities of code creation. However, comparing the variation in the software versions with the users' requests noted in the minutes of the meetings, we find that there is no fit between the code changes and the requirements. On the contrary the code reflected the original plan of the developers. This shows that the developers intentionally enacted a reification process where the users' feedbacks were not considered.

Boundary objects. During the project it was soon realized that the advanced mathematical technique, that the developers were using, required to be demonstrated to the users 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 was *visual* but it was too complex to be *usable* by the users. The testing phases were organized like presentations where the developers (instead of the users) operated the software and presented the results in form of numerical tables. The software was *up-to-date* but was simplified respect to the users' practices and changes to these simplifications were extremely difficult to implement. 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 use the software, they did not have a possibility to exchange their experiences.

Meaning negotiation in ISD: managerial processes and objects. The negotiation of meanings took place only partially. The components of the meanings negotiation were missing: after the initial phase participation became very scarce and brokering intermittent; reification was asymmetric and the prototypes, not having boundary objects' characteristics, reinforced the boundary instead of crossing it. Another example of artefact that failed its role as boundary object is the activity plan. The plan tried to draw an ordered process and it worked until major technical difficulties emerged. As the project became more complex, it was increasingly difficult to respect and the developers used the software to guide their actions planning their activities day by day. Within the hacker community it was no longer necessary to use a plan: they "simply know what to do".

Empirical Model

According to the previous analysis it is possible to identify the role of actors, social and managerial processes, objects and the relationships among them in an ISD project. In this section, focusing on the factors emerged from the case analysis, we propose a model that connects actors and objects through the process of negotiation of meaning (figure 5). This "atomic" schema, extended along the time dimension, fills the gap between managerial processes and objects considering the development as a whole where social actors and objects dynamically interact (figure 6).

From the case analysis we have observed that a crucial task in ISD is related to the negotiation of meaning between different actors (in the first phases between users and developers, in the late ones between 'hackers' and other participants). The starting point of the negotiation of meaning (figure 5) is the relationship that an actor (A1) establishes with individuals or members of other communities of practice (A2). In the starting phases SteelCo gave to the developers different artefacts (as activity models, production plans, previous software specifications) developed on the basis of its own

organizational experience and practices. Hence, in our model, A1 collecting and interpreting inputs and experiences from his specific environment, produces a cognitive representation: a boundary object (BO1). In details, at the beginning A1 has an embodied knowledge, not codified and strictly related to the specific environment and practices in which he has established his relations. A1's knowledge can be partially codified through the use of different formalisms. For example, A1 could *objectify*[4] some of his ideas in written documents or drawings, software code-lines, models, blueprints, etc. The subsequent relationship between A1 and A2 is mediated by objects and/or participation. If A1 and A2 belong to different communities of practice, their relation is crossing the boundaries of their own practice.

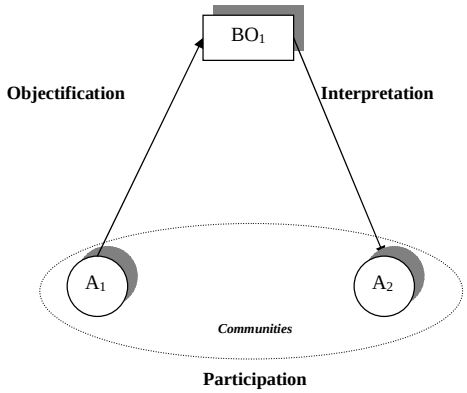

Figure 5 Basic Model of Social and Technological Interaction in an Activity System (Bolici, 2005)

The subsequent step is the interpretation process done by an actor (A2) interested in the specific artefact (BO1). This process is symmetrical and opposite to objectification. However A1 seldom codifies his whole knowledge in the artefact, and often part of his experiences remains tacit. Hence we can conclude that the interpretation of the object done by A2 could be very different from the original

[4] This concept is also used also in the Wenger's (1998) work referred to the reification process, but we prefer to use the word objectification to refer directly to the software.

concepts and the ideas of A1. This is important in order to realize that the simple exchange of an artefact (knowledge as an object, as recognized by some literature) is not enough to guarantee an effective knowledge transfer. Indeed, using a boundary object, an actor could engage in a negation of meaning with the author of the artefact sharing his cognitions and cooperating through the practices. These considerations emerged during the first phases of the project observing the interactions between the developers and the users. Developers participated into the users' practices trying to reach a better understanding of their activities and simultaneously some boundary objects (e.g. production plan) were used as keys element for the meaning negotiation between the two communities.

After that A2 interprets the artefact (BO1), several scenarios are possible (Fig.6): 1) A2 decides to cooperate in common practice with A1 and if from their cooperation another artefact (BO3) is created we observe dynamic interaction between reification and participation. This new object could include more than the common interpretation of BO1, also new elements rising from the comparison of different practices and cognitions. In the case study we observed this scenario when the activity models were created, starting from previous SteelCo models, as the result of the participation of Drew and Ted in the inventory practices; 2) A2 decides to re-elaborate BO1 though his own interpretation and his cognitive background creating BO2. This is the case of the programmers that starting from the users' requirements, elaborated different versions of the software simply relying on their own ability and interests; 3) A2 considers the artefact and its content useless, so simply starts again his search. In the SteelCo case this happened when the company refused to implement the software and continued the development on its own.

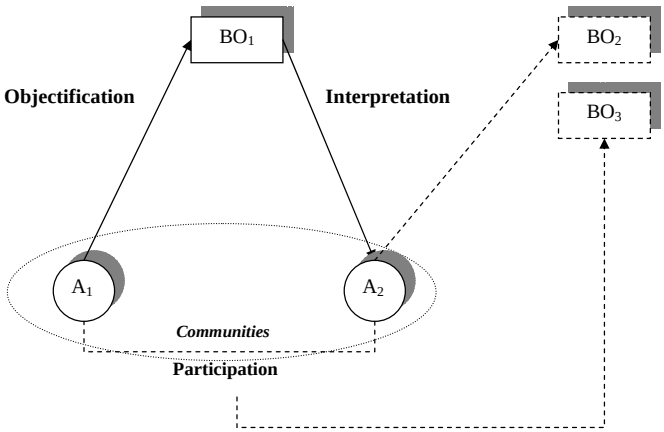

Figure 6. Interaction among two actors in meaning negotiating with possible outcomes

In the figure 7 we have represented the ISD case at SteelCo developing the model along the time dimension. At the beginning of the project (time 0) the two different communities exchanged their vision of the world through the dynamic interaction of reification (e.g. using activity models and the robot visual simulation) and participation processes (e.g. working together at SteelCo). Instead, in the following steps of the project, the actors involved in the ISD activities changed their way of working. We registered an increasing difficulty in communicating between the developers and the SteelCo's employees.

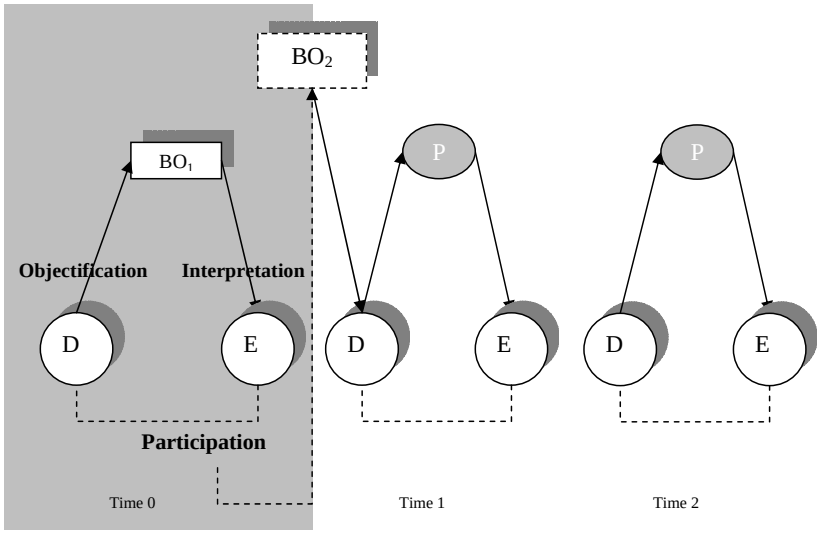

Figure 7. SteelCo analyzed through the Basic Model.

At time 1 and 2, the prototype's presentations made by the "hackers" in a mono-directional way became an occasion for showing the last functions developed (figure 7). At that stage, the developers (D) and the employees (E) were not sharing activities and practices anymore. The developers were continuing to follow their own ideas not interested in reaching a common understanding with the employees. On the other hand, the SteelCo employees, not directly involved in the process, diminished their active participation into it. According to these considerations, the prototypes -now marked with P instead of BO- showed in the last phases of the project can not be considered as boundary object since, as seen in the case, they obstructed the sharing of practices and perspectives across the communities' borders.

CONCLUSIONS

The goal of this article is to bridge the dichotomy between managerial processes and objects. The main contribution of this work is to make explicit the intertwined knowledge nature of ISD process and product and to show that with this view it is convenient to set knowledge creation as one of the goals of ISD.

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

We have presented participation and reification as key processes 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. With a vision of knowledge as something inseparable from practice, boundary objects as reificative means have been presented as particular instances of software that bridge the boundaries between developers and users.

Finally we have proposed an empirical model that shows the interlinked nature of participation and reification. The model highlights that objects and social processes are highly dependent from each other and both participate in the knowledge creation process as a crucial goal of ISD activities. Given these conclusions, the model may offer a unique and especially appropriate strategy for interpreting ISD according to the social-constructivist discourse which is becoming increasingly common today.

REFERENCES

Avison D.E., and Wood-Harper A.T., Vidgen R.T., Wood J.R.G. “A further exploration into information systems development: the evolution of Multiview2”, *Information Technology & People* (11:2), 1998, pp. 124-139.

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

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

Boland R. J., and Tenkasi R.V. “Perspective Making and Perspective Taking in Communities of Knowing”, *Organization Science* (7:4), 1995.

Boland R.J. “Control, causality and information system requirements”, *Accounting, Organizations and Society*, (4), 1979, pp.259-275.

Brooks, F.P “No silver bullet: essence and accidents of software engineering”, *IEEE Computer*, (20:4), 1987.

Brown J.S., and Duguid P. "Knowledge and Organization: A Social-Practice Perspective", *Organization Science* (12,2) 2001, pp. 198-213.

Carlile P.R. "A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development", *Organization Science*, July/August, 2002, pp.442-445.

Eisenhardt, K.M. "Buillding Theories from Case Study Research", *Academy of Management Review*, (14,4), 1989 pp.317-334.

Giddings, R. "Accomodating uncertainty in software design", *Communication of the ACM*, May, 1984, pp.428-434.

Jones, M., and Walsham, G. The limits of the knowable: organizational and design knowledge in system development. In Kendall, K. et al. (eds.), 1992, pp.195-213.

Lanzara, G. F., and Morner, M. "*The Knowledge Ecology of Open-Source Software Projects"* in the Proceedings of the 19th EGOS Colloquium, Copenhagen, 2003.

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

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

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

Nonaka, I. "A dynamic theory of organizational knowledge creation", *Organization Science*, (5:1), 1994, pp.14-37.

Polanyi, M. *Personal Knowledge: Toward a Post-Critical Philosophy*. Chicago University Press, Chicago, USA, 1958.

Polanyi, M. *The tacit dimension*, Peter Smith, Gloucester, USA, 1983.

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

Stage, J. "The use of description in the analyses and design of information systems", in Stamper et al. Collaborative Work, Social Communications and Information Systems, North Holland, 1991, pp. 237-260.

Standish Group, The Standish Group Report, http://www.scs.carleton.ca/~beau/PM/Standish-Report.html, 1995 (Visited 01.03.2005).

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

Strauss, A., and Corbin J. *Basics of Qualitative Research: Grounded Theory, Procedures and Techniques*. Sage Publications, Newburypark, 1990.

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

Wenger E. “Communities of Practice and Social Learning Systems”, *Organization* (7:2), 2000.

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

Yin, R.K., Case Study Research: Design and Method. Thousands Oaks, USA: Sage, 1994.