Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 2005 Proceedings

International Conference on Information Systems (ICIS)

December 2005

Information Systems Development as Inquiring Systems: Lessons from Philosophy, Theory, and Practice

Andrea Carugati
IESEG School of Management

INFORMATION SYSTEMS DEVELOPMENT AS INQUIRING SYSTEMS: LESSONS FROM PHILOSOPHY, THEORY, AND PRACTICE

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

Abstract

The paper presents a framework to interpret information systems development (ISD) as composed of activities for the creation and exchange of knowledge. The framework is based on the five inquiring systems presented by Churchman (1971).

The main contribution of the framework is in showing that in ISD the inquiring systems should be used complementarily: rationalism, empiricism, idealism, dialectic, and pragmatism are to be used by participants in ISD projects in accordance to the development phase they face. Since the framework focuses on the knowledge created during the interaction of individuals and groups (or the lack thereof) rather than on the specific development process, the framework is robust enough to be adaptable to different ISD methodologies. In the second part, the paper presents an ethnographic study of an ISD project. Through the analysis of the case material, it is evident that the participants display behavior very similar to the pure rational inquirer far from the plurality of approaches that the framework shows as necessary. This conclusion indicates that improvement in ISD has to be sought not only in methodological amelioration but in a combination of methodologies that support multiple inquiring systems and a change in the mindset of the actors involved through adequate practices.

Keywords: Information systems development, knowledge creation, inquiring systems

Introduction

Information systems development (ISD) can be conceived as the product of two interconnected activities: software producing activities and managerial functions (Abdel-Hamid and Madnick 1991). While this dichotomy is highly intuitive, current research tends to pair these two dimensions with a third one concerning knowledge creation. This branch of research focuses on knowledge issues in ISD either explicitly (see Avison et al. 1998; Markus et al. 2002; Richardson et al. 2001; Richardson and Courtney, 2004) or implicitly (see Beck 1999; Winter et al. 1995). Whether explicitly or implicitly, there is a trend toward a social constructivist approach to ISD where information systems and their requirements emerge from the interaction of multiple stakeholders' viewpoints (Iivari et al. 1998).

This paper continues this research tradition, presenting a theoretical framework that sets the spotlight on the knowledge creation and exchange aspects of ISD projects. In particular, the goal of the paper is to provide a framework to interpret information systems development in terms of the inquiring systems[1] proposed by Churchman (1971). The intentions are threefold. First, while methodologies are based on an overreaching ontology and epistemology (Avison et al. 1998; Iivari et al. 1998), focusing on micro-

[1]An inquiring system is a set of components that interact with the goal of understanding a situation, learning from it, and creating knowledge for the individuals in the system.

level ISD activities shows that multiple ways of creating knowledge are (or should be) used depending on the task and physical situation of the stakeholders. Second, it is necessary to understand which inquiring system(s) to use in any specific ISD activity to learn the most from it. The third purpose is to highlight the specific knowledge creation situations in ISD activities in order to provide managers with the appropriate means for their identification and management. The contribution of this paper is a framework to help managers see ISD activities as knowledge creation processes and to provide them with the principles for management in the hope of improving their chances of success with ISD projects.

Even if Churchman's inquiring systems are quite dated, using his work has several advantages. First, Churchman's inquiring systems are based on long-standing philosophical traditions that have been refined over time and, albeit biased by Churchman's interpretation of the philosophers' thoughts, are robust enough to be still valid today. This is proven by the extensive recent research that explicitly uses Churchman's work as departure point (e.g., Courtney et al. 1998; Kienholz 1999; Porra 2001; Swanson 1994). Second, inquiring systems have become an integral part of the information systems literature and as such their use eliminates the need to use ideas from reference disciplines (Richardson and Courtney 2004). Finally, this paper does not present Churchman's work as a way of perceiving reality but highlights, in a way familiar to the IS community, means to create and exchange knowledge during the different ISD activities.

This paper provides a brief introduction to Churchman's inquiring systems followed by a discussion focused on the creation of the framework. Since the interest is to provide a robust framework that can be used in conjunction with different methodologies, the framework will map the inquiring systems on basic ISD activities. Such activities are considered as building blocks whose specific sequence and content defines the ISD methodology used (Hughes and Wood-Harper 2000). In the second part, the paper presents an ethnographic study of an ISD project. The case is analyzed to identify the inquiring systems used by the developers during the project. The paper is concluded with a comparison of the framework with the case analysis. Conclusions are drawn on the effectiveness of ISD methodologies and management practices are presented to support the different knowledge creation situations.

Inquiring Systems and ISD

According to Churchman (1971), there are five basic modes of inquiry to create knowledge. These five inquiring systems[2] find their basis in the philosophies of Leibniz, Locke, Kant, Hegel, and Singer. In summary, these five modes are

- *Rationalism or the Leibnizian Inquiring System*: In this mode, the inquirer discovers through formal logic the truth governing the world. The system under study is considered as closed and governed by causal relationships. The inquirer creates networks of hypotheses, *fact nets*, and proceeds to build on these hypotheses until logic reveals a counterhypothesis that invalidates a part of the net. The Leibnizian inquirer is the isolated individual, the monad (Churchman 1971, p. 30), reasoning on the causal relationships governing the system that he wants to improve. The individual's logic becomes the guarantor of the truthfulness of the fact net. Pure structured methodologies (Boehm 1988) are built on the ideas of the Leibnizian inquiring system: once requirements are specified, the programmer proceeds on his own to develop the information system based on his internal logical reasoning. The software will, therefore, reflect the fact nets developed by the inquirer (i.e., the programmer).

- *Empiricism or Lockean Inquiring System*: This mode of inquiry gives more weight to data than theory. The inquirer is open to the environment and finds the truth in the external world. The inquirer is not isolated like the monad but participates in creating knowledge with a community of inquirers, the Lockean community. The inquirer uses data and the opinion of his community to explain the world. The community acts as guarantor of the truth through agreement and consensus. The Lockean inquiring system searches a single truth but there is no guarantee that two communities will agree on a common truth. In ISD, the groups of users and developers can be considered as two Lockean communities, where agreement can be found within a community but it is difficult to find it across communities.

- *Idealism or Kantian Inquiring System*: This mode of inquiry gives equal weight to data and theory. It recognizes that there are multiple ways in which a problem may be analyzed using multiple models that can be applied to the data. However the inquirer does not accept multiple truths; the objective is to find the ideal, and only, truth. The Kantian inquirer proceeds testing the fitness of a model to the data and creates knowledge by finding the model that best fits the data. In ISD it can be

[2]According to Churchman (1971, p. 43), a system is a set of components that interact with each other and with their environment to achieve a measurable goal stated by a client. A system can be designed by a designer under the supervision of a decision maker. Client, designer, and decision maker are roles and not people. Any player in a development effort can play any of these roles depending on the situation.

imagined that the Kantian inquiring system is used when developers search a model to mirror the client system into the software system. Many possibilities are tried and recursively eliminated until only one, the one with the best fit, the truth, remains.

- *Dialectic or Hegelian Inquiring System*: In this mode, the inquirer sees the truth emerging from opposing views. Debate between different worldviews (Churchman 1968; Linstone 1984; Orlikowski 1992) is seen as the only way to develop theses and antitheses to arrive at a synthesis that accommodates both worldviews. Hegel introduces the idea of multiple interpretations of reality even though he believed in a final "grand synthesis." Churchman (1968) envisioned two groups of people engaged in an ardent debate respectively defending their thesis and trying to destroy the other group's antithesis. The issue is then resolved by an impartial observer that synthesizes the arguments to create a synthesis. Synthesis generated from the impartial observer acts as the guarantor of the truth (Courtney et al. 1998). A good example of this inquiring system is the scenario approach at Shell, where groups with opposing views engaged in a dialectic inquiry and predicted the oil crisis in 1973 (Schoemaker 1995; Senge 1990, p. 178).

- *Pragmatism or the Singerian Inquiring System*: This inquiring system is based on the ardent debate of the Hegelian inquirer to create progress but accepts multiple sources of data and multiple interpretations of reality and consequently multiple truths. It is called pragmatic inquiring system because the truth produced is relative to the context and to the objectives of the inquiry. The multiple truths are found via an approach that continuously attacks currently held beliefs from multiple points of view. The world inquired is interpreted as an open system where all components might become part of the inquiry. The truth is temporary and contextually dependent because the more elements that are "swept in" (Courtney et al. 1998), the more the truth is likely to change. Progress is the ultimate goal of the system and it is measured quantitatively whenever possible, otherwise the groups' intuition that progress is made becomes the guarantor for qualitative measures (Richardson et al. 2001). In ISD, the use of the Singerian inquirer can be recognized in recent methodologies like the ones coming from soft systems methodology (Atkinson 2000; Winter et al. 1995), Multiview2 (Avison et al. 1998), and agile methodologies (Beck 1999; Fowler 2002). Richardson and Courtney (2004) propose a design theory for knowledge management systems based specifically on the Singerian inquiry. These methodologies let the information system emerge from the debate between developers and users, allowing multiple truths to be created pragmatically by the participants and not imposed top-down.

The traditional assumption is that the Singerian inquirer is more useful for wicked problems where objectives and methods to reach the objectives are unclear (Richardson and Courtney 2004, p. 2). This view, however, attributes the characteristic of the situation as existing outside the individual. Taking an interpretive approach (Orlikowski and Baroudi 1991), ill-structured and well-structured are not characteristics of the problem but of the observer, the result of individual and social construction. This is reflected in the fact that the way in which a problem is perceived by the individual changes according to the individual's context. From the individual perspective, it is, in principle, acceptable to use different inquiring systems according to the context rather than according to an externally defined characteristic of the problem. An individual working in isolation might tend to rely primarily on rationalism and idealism to move forward. Members of the same group will rely primarily on empiricism since members of a group possess a shared worldview (Boland and Tenkasi 1995) and the group can guarantee the goodness of their truth (Courtney et al. 1998, p. 7). Dialectic and pragmatism are inquiring modes used when two groups with different worldviews meet with the intent of learning from each other. Harrison and Bramson (cited in Kienholz 1999) suggest, however, that individuals—despite being flexible—display favorite modes of inquiry and that this flexibility is not necessarily always used.

The following part of the paper maps the inquiring systems on the ISD activities. The inquirer types will be used in the analysis of the case material to evidence harmony or disharmony between the inquiring systems in the framework and the behavior displayed by the developers.

Framework Development

The ISD[3] process can vary radically depending on which development methodology has been chosen. However, there are five basic activities that are shared—albeit with different names—by most methodologies (ISO 2000; see also the references to ISD methodologies below): (1) *identification and concept*, (2) *requirements definition*, (3) *system design*, (4) *implementation*, and (5) *testing and operation*.

[3]An information system is intended here à la Churchman: a system that processes information in order to increase the probability of attaining the (larger) objectiver (Porra 2001, p. 23). With this definition, IS can be as simple as a software package or as complex as a socio-technical system depending on the definition of its boundary and of the inquiring system considered.

These activities can be performed in various sequences, recursions, and revisions, can be omitted, or focus more or less on the customer organization (Avison et al. 1998, p. 126). According to the specific sequence adopted, different ISD methodologies can be recognized: waterfall (Boehm 1988) if the activities are performed in sequence, spiral (Boehm 1988) if the activities are performed cyclically, rapid prototyping if requirements definition is in fact a fast cycle through the remaining activities (Beath and Orlikowski 1994; Martin 1991), agile methodology if attention is given to collaboration, speed, and cyclicality (Beck 1999; Lindstrom and Jeffries 2004). A methodology like Multiview includes the first two activities but with a major techno-social scope, whereas Multiview2 includes all activities specifically including organizational and sociotechnical analysis in requirements definition (Avison et al. 1998). Therefore, in this paper, the five ISD activities are considered as building blocks of ISD methodologies.

Interpreting these five activities in terms of systems for creating knowledge, it is possible to link them to the use of one or more inquiring systems. The following sections maps the inquiring systems on the ISD activities taking into consideration the goal of the activity, its progress, the active roles and the context (e.g., the physical situation) in which the activity is carried out.

Identification and Concept

As a norm, this activity is carried out within a company that experiences problems or sees opportunities that it wants to seize with the use of information systems (Hughes and Wood-Harper 2000). This group will be referred to as the "users."[4] The users discuss the situation and how they expect an information system to help them improve it. Users are usually not expert developers and, therefore, they can conceive some functionalities for the information system but neither the details nor the additional features that developers can create for them with technology.

The most likely inquiring mode through which the group creates knowledge is empiricism: they will consider the world as an open system and try to find the data to support solutions in the environment. Consensus within the group acts as a guarantor for the created knowledge.

Requirements Definition

This activity is carried out in cooperation between users and systems developers with the scope of specifying the information system (Sommerville and Sawyer 1997). Functional requirements emerge from a debate between users and developers with the purpose of reaching an agreement on the goals of the project. In this phase, the groups should behave as Singerian inquirers. The two groups debate ideas about problems and possibilities. They will tend to include in the discussion elements from all interconnected systems and they will reach a consensus on the functions of the information system. When using this inquiring system, ideas are not necessarily shared by every individual. This is known and contingency plans for rework can be made in anticipation. The phase is over when enough progress is made. Progress can be measured by changes in the requirements document (Sommerville and Sawyer 1997), in the use cases (Martin and Odell 1992), or in the user stories (Fowler 2002) depending on the ISD methodology adopted.

Design

During the design phase the developers work as a group to plan how to bring the requirements to life. A variety of technical decisions is taken (e.g., about the programming language and style, modularization, interfaces, etc.). These decisions usually allow developers to work independently during the coding activities. During this phase, the developers add to the knowledge obtained from the users, creating their own hypotheses about the users' system and the solution requested. In this step, knowledge is created through interpretation of the information gathered from the users filtered through previous programming experiences.

The development group behaves as a Lockean inquirer. It is open to the environment but creates the truth, the knowledge of how the new system has to operate, within the group. Like before, the members seek consensus within the group that acts as a guarantor for the created knowledge.

---

[4]*Users* here is a simplification of Churchman's (1971, p. 43) notion of *client* as one of the main roles in system design. While users might have multiple interests, for the scope of this paper, a monolithic view of users is sufficient because, with respect to IS developers, they would look like a compact community.

Implementation

In this activity, the developer, now isolated, begins to elaborate through his knowledge what he learned from the users and from his colleagues. In the process of coding, the developer interprets the information in his possession and creates, consciously or unconsciously, a network of hypotheses (fact nets) that bring him from ideas to actual code. The system developer acts as a monad in a Leibnizian inquiring system, creating new knowledge through logical deduction. Fact nets are created based on hypothesis on the users' system. The fact nets grow as more and more elements are added into the software.

Occasionally the developer faces uncertainty regarding how well a particular algorithm replicates the users' system. If multiple algorithms or models have to be compared, the developer enters a Kantian mode of inquiry, trying to find which algorithm fits best the users' system. In this inquiry, the developer might follow his own judgment or search for answers outside. Even though the answers come from the outside, the goal of the search is to justify the use of a certain model, not to put under discussion the work previously done. The search ends when a model is found that fits the particular situation.

The result of the knowledge work during design and implementation is a continuous evolution in the developers' knowledge. These activities are not simply an operationalisation of the requirements agreed with the users but are the arena for criticism, suggestions, logical development, and experimentation. These are all processes that contribute to the creation of software artifacts that can be different from what the users expect.

Testing and Operation

In this phase,the code, in form of finished product, prototype,or release, is shown to the users for testing. Both users and developers should be present during the operation phase. The users operate the software and compare the experience with their expectations. Expectations are debated to achieve new agreements; he Singerian inquiring system is used again. If the software is implemented, the users have the opportunity to use it for regular work and can request additional features or modifications at the following meetings. New knowledge created by the users during operation is outside the *modus operandi* of the debate and is created using the Lockean inquiring mode.

The framework is presented in Table 1. Each activity (vertical axes) is mapped on the inquiring systems (horizontal axes) that are desirable to be active during the activity.

For each activity, the most effective inquiring system has been indicated considering the goals and the physical constraints of the activity. In particular, even though a methodology based strongly on the Singerian inquiring system might be more effective (Richardson et al. 2001), the debate required by this inquiring system is not possible nor is it desirable for all activities.

Given the content of the framework, the second part of the paper will to present the results of an ethnographical investigation of an ISD project. The scope is to highlight the reasons for the inevitable disagreements between the framework and the complex nature of reality and propose ways to find conciliation.

Case Study

The case reflects the observation of the activities of a team responsible for the development of an information system based on combinatorial optimization (CO). The objective of the ISD project was to develop software that could aid decision making in planning and control tasks in the inventory department of a manufacturing company. The complexity and novelty of the use of the CO technique to solve planning problems was such that university researchers were required to carry out the development headed by a project manager from an information technology consulting company. Four universities each contributed two team members, one with a supervisory role and the other with practical functions in the project. The manufacturer (SteelCo[5]) contributed four team members: two sponsors, an area manager, and an internal project manager. The consultancy (Consult) contributed with the project manager. Of the total 13 team members, 7 took an active part in the development process. The seven

[5]The names of the companies and the project participants have been disguised.

Table 1. Theoretical Framework				
ISD Activities ↓	**Inquiring Systems Active During ISD Activities**			
Identification and Concept		Activity carried out by the information systems stakeholders in the user's organization.		
Requirements Definition				Activity carried out by users and developers. Achievement of agreement cannot be postulated. Groups and individuals might bring home different views of the truth.
Design		Activity carried out by the development team.		Activity carried out by both teams.
Implementation	Activity carried out by the individual developer. Fact nets are created based on internal logic. New knowledge is created outside the control of other stakeholders.		Activity carried out by the individual developer. When in doubt the developer tries multiple models until the best fit is found.	
Operation		If the information system is put in operation, testing can occur in real working situations. New ideas are generated among the users to be included in requirements.		Phase carried out by users with developers' supervision. New ideas generated from debate are included in the requirements. Progress in this phase is the guarantor.
Inquiring Systems →	Rationalism (Leibnitz)	Empiricism (Locke)	Idealism (Kant)	Pragmatism, Dialectic (Hegel, Singer)

were, specifically, the overall project manager (Ted), the developer of the planning part of the software (Joshua), the developer of the control part of the software (Walter), the person in charge of system modeling (Drew), the person in charge of the organizational change process (Tom), and the sponsors for the client company (Rob and Jim). Even though some of the supervisors knew each other before the project started, the parties with executive tasks had never met before the project started. The team was to work together on a system development for a period of 3 years.

The team members were located in different areas. Travel activities were kept to a minimum. Even though some team members were situated rather close to each other, they did not meet more often than the other team members.

The project had two characteristics that made it an interesting example for the argumentations presented in this paper. The first characteristic was that the information system was based on a mathematical technique whose properties were new to the users. The second characteristic was that the development team was external to the client, with the developers working from distributed locations on tasks specific to the developers. Both characteristics point at knowledge creation and exchange as key factors for the success of the project. The first characteristic made it important for the users to learn about the mathematical technique in order to be able to specify the requirements; the second made it important for the developers to learn from each other about their work.

Research Methodology and Data Collection

The case analysis is the result of an iterative process of conceptualization grounded in theory, collected data, and reflections. This iterative process is in line with my ontological belief in a reality that is not external and independent from the individual but is subject to continuous interpretation (Orlikowski and Baroudi 1991) and is the result of a process of social construction (Berger and Luckman 1989). A key advantage of this approach is that equivocal outcomes are potentially accounted for by understanding how the social meanings attached to a project activity originate and evolve (Barrett and Walsham 1999).

The empirical part of the paper is based on a 3-year-long interpretive case study. During the project, I performed activities of liaison between IS stakeholders in the company and the development team. The participative nature of my work in the project has been a means to observe closely the development project. In this period, I gathered data in the form of observations, interviews, transcripts and minutes of project meetings, and software versions.

Toward the end of the project, I performed in-depth semi-structured interviews with the team members[6] where the interviewees were asked to talk about their view of the history of the project, their impression on how the project was managed, and their roles and activities within it.

After the project was concluded, I convinced the most active participants to give me all of their e-mail files for the development period. It was important to ask for the e-mail files after the project because this would avoid bias in the content, workarounds, or opportunistic selection. The creation of the CD of e-mail files was done under my supervision, assuring that no tampering occurred. The e-mail files were an interesting data source because they contained information of direct communication among team members and helped to support field notes and interview content. The e-mail files are also probably the most informative documents because, as the team was distributed, most of the communication was electronic. The e-mail files give a very precise story of the project because each e-mail heading includes a time and date stamp, the sender and recipients, and information about attachments. From this information it is possible to infer who is active at a particular time and for what purpose. The use of the "reply" button allows discussion threads to be followed, which can be used to understand the themes considered most important by individual project members (those who kept the thread going). Furthermore, thanks to the reply button, having the e-mail of four of the seven main players[7] warrants good coverage of the bulk of the electronic communication.

The analysis was informed by my focus on the ISD activities and on the use of the inquiring systems. I did the coding following two consecutive cycles. First, I reviewed the empirical material to identify the activities carried out by the developers. This process was carried out three times: once for my field notes, once for the interviews, and once for the e-mail files. Then, for each activity, I recoded the data to look for inquiring systems used. Again the process was carried out three times. The intention was to find overlapping or disproving evidence across the data sources and to ultimately present an accurate case story. The main reason for repeating the two coding cycles on the three data sources was to grant the informers complete confidentiality, so I could not ask another coder to check for the validity of my observations. The use of three sources of data acted as a surrogate for internal validity.

The Case Story

In the following sections, a simplified view[8] of the ISD activities that unfolded during this project are presented with special focus on the inquiring systems adopted by the developers.

---

[6]I interviewed all but two team members. One team member changed jobs; the other abandoned the project 6 months into the development.

[7]As noted earlier, one team member changed jobs and another abandoned the project very early in development. A third was using an e-mail system not accessible from outside his organization.

[8]Projects of this size and length are much more complex than the simple view provided here. However, an exhaustive report of the case is not appropriate in this paper. This section aims to present a few instances of the activities discussed to show that multiple inquiring systems are active in real cases.

Identification and Concept

This activity was carried out by the client. The client had some experience with the use of the mathematical technique to solve complex planning problems and decided to start a larger project aimed at developing integrated scheduling and control software to manage the activities of the inventory and production lines. This phase was carried out by expert project managers of the company without consulting the developers. The results were included in a project description brief that was later used as a starting point for the requirements definition phase. The project managers arrived at the decision to start the project working as Lockean inquirers: as a group they decided that the idea was favorable.

Requirement Definition

This activity was mostly done by the developers alone. The developers were taken around the factory for an afternoon and then were stationed for 2 weeks at the users' workstations to observe their work. Based on these observations, the developers were required to create a requirements document. Only two of the three developers worked on the requirements document and they did so individually. The discussion about the requirements happened on the train on the way to SteelCo. The other developer—Walter—did not even consider participating at this stage. He commented, showing a typical rationalist approach,

> *The primary goal for me is to have the control software functioning in a real system and to get an idea of the problems with data collection and processing… and I took the decision to work on this other task because it was well defined and it was just ready to be started.*

In hindsight the other developer—Joshua—recognized this situation as unsatisfactory, although his comment reflects a firm anchoring in the rational way of acting:

> *I think that it [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 they [SteelCo] did not know what we needed and we did not know either.*

The requirements document was presented to the users in a meeting where there was no debate and the developers had the freedom to do basically whatever they pleased. The creation of the document was based on the use of rationalism, where not even the developers agreed on the content, rather than based on the Singerian inquirer, where multiple perspectives should have collaborated to make the requirements emerge. This situation was also made possible by the attitude of the project sponsors at SteelCo, who did not intend to participate in the initial stages. For example, when invited to a meeting for deciding the initial phases of the case, Rob wrote,

> *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.*

Design

The design activity was exclusively carried out by the development team. In this phase, the developers decided that the software had to be divided in modules; they decided about processes for material handling; they decided about use of overtime. All of these decisions were made thinking that they were either logical or possible and therefore good. These decisions became the hypothesis at the base of the fact nets, the networks of causal relationships that characterized the software. The inquiring system used here was again the rational one. The developers—Walter had begun to actively participate in the project at this point—worked separately on the design of their modules. The few voices that spoke against taking decisions without consulting the client were ignored and even marginalized. The customer was never consulted about these decisions because it was thought that comments would eventually emerge during the testing of the systems modules. Once Ted, the project manager, had made the plans and defined the interfaces between modules (that were never used), he stopped being actively involved in the project, showing himself a firm believer in the rational mode of carrying out projects. This behavior is best shown in the profile of the e-mail exchanged during the period January 2000 to December 2001 (Figure 1). The lines in the graph represent the e-mail sent by each team member over the period. To make the graph easier to read, the e-mail of Joshua, Walter, and Drew has been grouped.

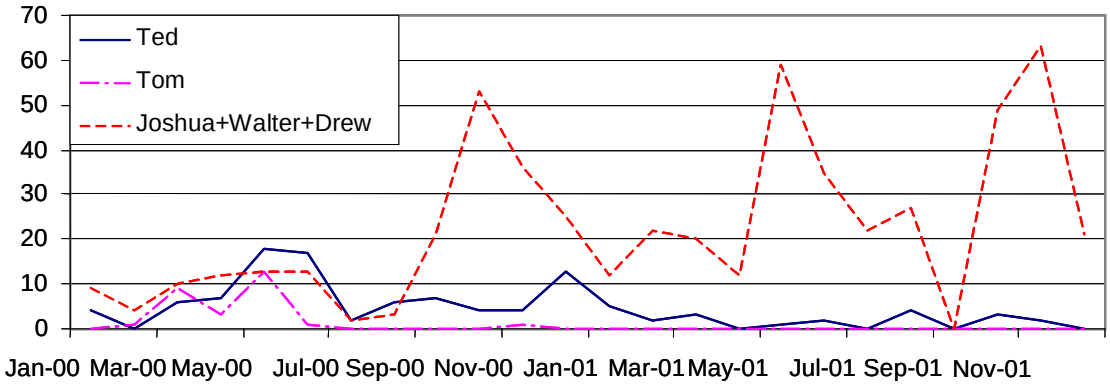

Figure 1. E-Mail Exchanged During the ISD Project (2000 and 2001)

The e-mail clearly shows the disappearance of Tom after July 2000, resulting from his disagreement with the way the project was advancing. In the period after September 2000, the e-mail sent by Ted dropped dramatically even though the developers were going through the design phase, and later the implementation phase, where they needed a lot of help. Joshua commented on this behavior, displaying both appreciation and regret for it.

> *In a way it was a relief [when Ted stopped acting as project leader] because it became easier to set a course for what to do. I felt really that I was sitting between Walter and Ted. Walter could not really understand what Ted was talking about and I could only partially understand. And that situation could not continue. We could not be in a situation where one talks on one level and one of the participants completely could not understand what was going on.…I think however that it was unfortunate that he disappeared completely…we could have used some help from the most experienced of us.*

In conclusion, the design phase was characterized by a rationalistic approach both for the developers and the project manager. The discussions that seem to bring agreement were not transferred in practice into harmonized and coherent software modules.

Implementation

The implementation activity—the coding of the software—was very problematic. The complexity of the mathematical technique made the code very difficult to implement. During this phase, the developers worked mostly alone, like Leibnizian inquirers, developing their own modules on the hypothesis obtained in the previous phases and further developing these hypotheses on an individual basis. For example, one developer arrived at the decision that for testing purposes it would not be necessary to consider machine breakdowns. This decision was logical at the time: details were less important than speed. The developers thought that the validity of this simplification would be discussed later during the testing phase. In this phase, the developers also suffered from the project manager's decision to step back although they did not engage actions to make him participate more. As Joshua said,

> *Ted was not the kind of leader that would ask us how the situation was going and call for meetings and so on. He expected us to go to him and tell him how it was going. And what happened in reality is that we let him out of the loop.*

The developers also used idealism as a mode of inquiry to a large extent. For example, the modeling of the arrival of material or the modeling of the distribution of delays emerged from a process of data collection and research for the model with the best fit (usually a statistical distribution). Similar to the other simplifications, the developers thought that the users would validate the use of these models during testing. During coding, the developers behaved as expected in the framework (Table 1).

Testing and Operation

During the development project the development team held three demonstrations of the software during the 16th, 19th, and 21st months of the project. The idea behind testing 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 used the system and showed it to the users. Not knowing exactly what they were observing, the users were not able to sustain the ardent debate required to create knowledge in a Singerian inquiry. The developers firmly led the discussion, impeding the ability of the users to criticize the validity of the simplifications in the software. There was very limited knowledge creation that would have brought the project closer to being successful.

This project joined the other 84 percent of ISD projects that fail each year.[9] The software was never implemented and when, approaching the end of the project, I asked the company project managers about their acquired knowledge 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 they also failed to create knowledge for approaching similar problems in the future. However the project sponsor at SteelCo was aware of the problem and made a pointed comment about the developers' behavior that indicated precisely the location of the root of the problem.

> *We felt that some people withdrew to their ivory tower, and cultivated their own interests there, where it perhaps could have been more advantageous for them and us if we had more dialogue in that period.* (Rob)

The similarity between retiring to the ivory tower and the *modus operandi* of the rational inquirer is very well pointed out. Joshua was also sympathetic with this problem. He said,

> *What bothers me enormously, now that the project is finished, is that there has been so little control over the situation. It hasn't worked as a project, it has worked as individual persons, each with their goals, and who did their little thing, and then we have tried our best to lead it to something that made sense. Quite a lot of things didn't work because the developers had to communicate to integrate their parts of software, a detail which hadn't been thought into the project from the beginning – and this ended up causing us a lot of problems.*

Case Analysis

The activities carried out by the developers show that the main inquiring system used was rationalism. Rationalism is used consistently throughout the project and where other behaviors emerge, the people bringing them forth are marginalized immediately. Table 2 presents an overview of the inquiring systems active for the developers during the different activities of the project. The DESIRED inquiring system from Table 1 are presented in block letters.

As it can be seen in Table 2, it is only in the implementation activity that the displayed behavior and the desired behavior overlap. This overlap seems to be the result of the predilection of the developers for the rational inquiring system rather than the result of a reflection process.

While it is not possible to link the lack of overlap to the failure of the project (failure could derive from a combination of multiple factors including the new mathematical technique, inexperienced programmers, or distributed team members) it can at least be evidenced that both the developers and the clients were dissatisfied with the way in which knowledge was exchanged during the project. This dissatisfaction is not value-free since knowledge of how the project was going could at least have allowed the project managers to pull the plug or take other drastic measures. The analysis can be continued at two levels: activity and global.

At the activity level, the analysis shows that the developers followed mainly a single mode of inquiry even where other modes would seem more appropriate. This is in line with the results of Kienholz (1999), who found that people have a predilection for a particular inquiry. Nonetheless, managers have been known to be able to influence the way in which people interact to create knowledge. Mitroff et al. (1977) and Linstone (1984) show that, given the right preparation and context, people used to rational inquiry can work as Singerian inquirers. The *laissez faire* style of the project manager not only left the developers to their own devices but also did not guide them through the delicate process of adapting their inquiring style.

---

[9]Numbers provided by the Standish Group (see **www.standishgroup.com/**; accessed in August 2003). The report is available at **http://www.standishgroup.com/sample_research/chaos_1994_1.php**.

Table 2. Practice Framework for the Developers				
ISD Activities ↓	**Inquiring Systems Active During ISD Activities for the Developers**			
Identification and Concept		**DESIRED**		
Requirements Definition	Activity carried out by developers singly. Users passive.			**DESIRED**
Design	Activity carried out by developers singly. No discussion about coordination in the initial phases of the project.	**VIABLE**		**DESIRED**
Implementation	**DESIRED** Activity carried out by the individual developer. New knowledge is created outside the control of other stakeholders.		**DESIRED** Activity carried out by the individual developer. When in doubt the developer tries multiple models until the best fit is found.	
Testing and Operation	The phase of testing is carried out by developers with the users as observers. Debate is weak; new knowledge does not filtrate from users to developers or vice	**DESIRED**		**DESIRED**
Inquiring Systems →	**Rationalism**	**Empiricism**	**Idealism**	**Pragmatism and Dialectic**

At the global level, this study shows that, despite good rational reasons for promoting Singerian-type inquirers in all ISD activities, there is a gap between the theory and the practice and this gap is part of normal, traditional ISD practices and therefore very difficult to close. One way of proceeding toward a wider adoption of the Singerian inquirer is through continuous utilization of this practice until it becomes a habit. The Singerian inquirer requires cooperation and no locus of authority within the systems; authority should be diffused in every participant (Richardson et al. 2001). Therefore, project managers have to find the right equilibrium between managing and coaching if they want both the right inquiring system used and the right channels open between users and developers.

Normative Implications and Limitations of the Framework

Viewing ISD as a process of knowledge creation provides a different agenda for what should be done to create relevant systems. As expressed by Wenger (2000, p. 244), with this view, informal processes like conversations and brainstorming become the primary source of value creation while formal designs are still important but contribute to value creation to the extent that they serve knowledge processes. This section provides normative implications on process structuring and managerial activities deriving from this way of interpreting ISD.

Process Implications

Speed and Size. Developers will tend to develop the software partially from the discussion with the users and partially from their own reasoning. If this knowledge results in small fact nets, they can be easily analyzed and invalidated during testing. However, if the nets are very large, discovering all of the causal relationships that have been embedded is almost impossible. Time intervals between testing activities should be kept short to limit the uncontrolled growth of fact nets. Consequently, the size of the development task should be controlled so that small prototypes or modules can be developed in the short time allocated.

Cyclicality. In the identification phase and in the design phase, consensus is achieved in different groups. Therefore, there is no *a priori* guarantee that the system will correctly serve the users. However, the more the Singerian the inquiring system used, the more likely it is that the resulting information system will serve all involved interests. In process structuring, this translates in planning to going through the testing/operation phase many times, causing developers and users to debate the work done according to the Singerian inquirer.

Managerial Implications

Support the Singerian Inquirer. Entering into the Singerian inquiring mode is not easy. A presentation, like in the example case, is not a debate. The Singerian inquirer requires cooperation and no locus of authority, which should instead be shared by every participant (Richardson et al. 2001). Project managers on both sides should agree to mediate the discussion instead of leading it and should structure it so that all necessary elements are swept in. Managers are also the guarantors that ethics are upheld (i.e., the client is served).

Live testing. In ISD projects, most of the work on the code is done by developers. They know the code inside out and assumptions become easily black boxed, in other words, developers cannot attack with critical spirit their own fact nets because these have emerged from the developers' own logic. Therefore, users should be able to personally test the information system in order to use the knowledge gained during testing as basic for debate (Carugati 2004). Project managers should agree on a formal requirement that prototypes or modules be usable by the users during testing in an operational context. Anything less than that spells failure in the long run.

These guidelines do not reject existing ISD methodology practices but rather add new practices. For waterfall-type developments, the takeaway would be to speed up development and to identify and develop working modules that can be put into operation within short delays. For spiral-types developments, the takeaway would be to go through the cycle rapidly before the fact nets become too large. For rapid prototyping the attention is on creating the right condition during testing so that the Singerian inquiring system can operate effectively. Finally, these guidelines provide theoretical support for the practices advocated in agile methodologies.

One limitation of the framework is that its development is based on theoretical reasoning rather than from a case in which the framework was successfully applied. The framework needs to be tested in an action research setting in order to see the extent to which it can help, in practice, the ISD process. The author plans to carry out this type of study in the future but hopes that this paper will stimulate others to pick up the challenge of uncovering the knowledge dimension of ISD.

Conclusions

This paper presents a view of ISD where the software creating activities and managerial activities are joined by knowledge producing activities. The paper shows that it is possible and advantageous to map the basic ISD activities on the inquiring systems presented by Churchman. The paper shows that, despite the recent focus of researchers on the Singerian inquiring systems as the superior system for knowledge creation, human nature and physical constraints prevent the application of this inquiring mode in all ISD activities. As a result, the different inquiring systems can at best be active in a scattered fashion and provide diversified ways of creating knowledge which in turn have to be supported by multiple and adequate practices.

The case study presented shows that, in reality, personal habits and practice traditions make the inquiring system used converge on a single mode of inquiry, the rationalist mode, diverging even further from the methodologies based on the Singerian inquirer.

The paper concludes with the presentation of a series of normative recommendations on how to structure the ISD process and on the managerial activities required as a consequence of applying a knowledge perspective to ISD.

References

Abdel-Hamid, T., and Madnick, S. *Software Project Dynamics: An Integrated Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Atkinson, C. J. "The Soft Information Systems and Technologies Methodology (SISTeM): An Actor Network Contingency Approach to Integrated Development," *European Journal of Information Systems* (9), June 2000, pp. 104-123.

Avison, D. E., Wood-Harper, A. T., Vidgen, R. T., and Wood, J. R. G. "A Further Exploration into Information Systems Development: The Evolution of Multiview2," *Information Technology & People* (11:2), 1998, pp. 124-139.

Barrett, M., and Walsham, G. "Electronic Trading and Work Transformation in the London Insurance Market," *Information Systems Research* (10:1), January 1999, pp. 1-22.

Beath, C. M., and Orlikowski, W. J. "The Contradictory Structure of Systems Development Methodologies: Deconstructing the IS-User Relationship in Information Engineering," *Information Systems Research* (5:4), December 1994, pp. 350-377.

Beck, K. *Extreme Programming Explained: Embrace Change*, Addison Wesley, Reading MA, 1999.

Berger, P. L., and Luckman, T. *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*, Anchor Books, New York, 1989.

Boehm, B. "A Spiral Model of Software Development and Enhancement," *IEEE Computer* (21:5), May 1988, pp. 61-72.

Boland, R. J., and Tenkasi, R. V. "Perspective Making and Perspective Taking in Communities of Knowing," *Organization Science* (7:4), 1995, pp 350-372.

Carugati, A. *Multiple Perspective Approach for the Development of Information Systems Based on Advanced Mathematical Models*, unpublished doctoral dissertation, Technical University of Denmark, 2004.

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

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

Churchman, C. W. "The Role of Weltanschauung in Problem Solving and Inquiry," internal working paper No. 94, Space Science Laboratory, University of California Berkeley, 1968.

Courtney, F. J., Croasdell, D. T., and Paradice, D. B. "Inquiring Organizations," *Australian Journal of Information Systems* (6:1), September, 1998, pp. 3-15.

Fowler, M. "The New Methodology," 2002 (available online at **www.martinfowler.com**; accessed June 2003).

Harrison, A. F., and Bramson, R. M. *The Art of Thinking*, Berkley, New York, 1982.

Hughes, J., and Wood-Harper, T. An Empirical Model of the Information Systems Development Process: A Case Study of an Automotive Manufacturer," *Accounting Forum* (24:4), December 2000, pp. 391-406.

Kienholz, A. "Systems Re-Thinking: An Inquiring Systems Approach to the Art and Practices of the Learning Organizations," *Foundations of Information Systems*, February 1999 (available online at **http://www.bauer.uh.edu/parks/fis/inqre2a1.htm**; accessed March 2005).

ISO Standard 15704, "Industrial Automation Systems—Requirements for Enterprise-Reference Architectures and Methodologies, International Standard Organization, Geneva, Switzerland, 2000 (available online at **http://www.mel.nist.gov/sc5wg1/gera-std/15704fds.htm).**

Iivari, J., Hirschheim, R., and Klein, H. K. "Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Informaition Systms Research* (9:2), June 1998, pp. 164-193.

Lindstrom, L., and Jeffries, R. "Extreme Programming and Agile Software Development Methodologies," *Information Systems Management* (21:3), Summer 2004, pp. 41-52.

Linstone, H. A. *Multiple Perspectives for Decision Making*, North-Holland, New York, 1984.

Markus, M. L., Majchrzak, A., and Gasser, L. "A Design Theory for Systems That Support Emergent Knowledge Processes," *MIS Quarterly* (26:3), 2002, pp. 179-212.

Martin, E. J. *Rapid Application Development*, Macmillan, New York, 1991.

Martin, J., and Odell, J. *Object-Oriented Analysis and Design*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

Mitroff, I. I., Barabba, V. P., and Kilmann, R. H. "The Application of Behavioral and Philosophical Technologies to Strategic Planning: A Case Study of a Large Federal Agency," *Management Science* (24:1), September 1977, pp. 44-59.

Orlikowski, W. J. "Learning from Notes: Organizational Issues in Groupware Implementation," in *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*, M. Mantei, and R. M. Baecker (Eds.), Toronto, Ontario, Canada, November 1-4, 1992, pp. 362-369.

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

Porra, J. "A Dialogue with C. West Churchman," *Information Systems Frontier* (3:1), 2001, pp. 19-27.

Richardson, S. M., and Courtney, J. F. "A Churchmanian Theory of Knowledge Management System Design," in *Proceedings of the 37th HICSS Conference*, R. Sprague (Ed.), Computer Society Press, Los Alamitos, CA, 2004 (CD-Rom).

Richardson, S. M., Courtney, J. F., and Paradice, D. B. "An Assessment of the Singerian Inquiring Organizational Model: Cases from Academia and the Utility Industry," *Information Systems Frontiers* (3:1), 2001, pp. 49-62.

Schoemaker, P. J. "Scenario Planning: A Tool for Strategic Thinking," *Sloan Management Review* (36:2), Winter 1995, pp. 25-40.

Senge, M. P. *The Fifth Discipline*, Random House, London, 1990.

Sommerville, I., and Sawyer, P. *Requirements Engineering: A Good Practice Guide*, Wiley, New York, 1997.
Swanson, E. B. "Churchman's Theory of Design Integrity," *Interfaces* (24:4), July 1994, pp. 54-59.
Wenger, E. "Communities of Practice and Social Learning Systems," *Organization* (7:2), 2000, pp. 225-246.
Winter, M. C., Brown, D. H., and Checkland, P. B. "A Role for Soft Systems Methodology in Information Systems Development," *European Journal of Information Systems* (4:3), 1995, pp. 130-142.