On the Use of Inquiring Systems in Information System Development Activities: Lessons from Philosophy, Theory, and Practice

Andrea Carugati
IESEG School of Management

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 framework shows that the inquiring systems: rationalism, empiricism, idealism, dialectic, and pragmatism should be used by the participants in ISD projects in accordance with the development activity they face. Since the framework focuses on the knowledge created during the ISD activities rather than on the development process, the framework is robust enough to be adapted to different ISD methodologies. In the second part the article presents an ethnographic study of an ISD project. Through the analysis of the empirical material it is shown that the participants display a behavior very similar to the pure rational inquirer far from the plurality of approaches that the framework shows as possible and necessary. The implications for practice and theory are that improvement in ISD has not only to be sought in methodological amelioration but in a combination of methodologies that support multiple inquiring systems and a change in the mindset of the actors through adequate practices.

Key-words:
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 (e.g. see: Avison et al., 1998; Richardson et al., 2001; Richardson and Courtney, 2004; Markus et al., 2002) or implicitly (e.g. see: Beck, 1999; Winter et al., 1995). Whether explicitly or implicitly there is a trend towards a social constructivist approach to ISD where information systems and their requirements emerge from the interaction of multiple stakeholders’ viewpoints (Hirshheim et al., 1998).

The above mentioned literature points out that using alternative viewpoints can enhance the chances for ISD success focusing on the use of a fundamental knowledge creation philosophy that permeates the entire ISD process. However a good understanding of the internal dynamics of low-level activities is important as they are the building blocks whose specific sequence and content defines the ISD methodology used (Hughes and Wood-Harper, 2000). Methodological research typically remains at the level of the complete process and does not focus on the activity level. However the actors involved in ISD find themselves continuously swinging between long term planning (process) and day-to-day work (activity) and have to manage both. So, while a certain philosophy might be good “in general” for the entire process it does not mean that it is good for every activity in the process.

This paper tries to fill this gap presenting a theoretical framework that sets the spotlight on the low-level ISD activities and their knowledge creation and exchange aspects. The intentions are
threelfold. First, focusing on micro-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, the paper shows that it is necessary to understand the implications of the use of a particular way of creating knowledge in any specific ISD activity. 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.

In order to have a reference point for the variety of knowledge creation situations the paper presents a theoretical discussion that leads to mapping information systems development activities on the inquiring systems\(^2\) proposed by Churchman (1971). The first contribution of this article is therefore a framework to help managers to see ISD activities as knowledge creation processes and to provide them with the principles for their management in the hope to improve the chances of success of ISD projects.

Even if it could be argued that Churchman’s inquiring systems are quite dated (1971) using Churchman’s work has several advantages. First, the Churchmanian inquiring systems are based on long standing philosophical traditions that have been refined over time and therefore, albeit biased by Churchman’s interpretation of the philosophers’ thought, are robust enough to have stood the test of time. This is proven by the large and recent amount of research that explicitly uses Churchman’s work as departure point (e.g. Kienholz, 1999; Courtney, Croasdell and Paradice, 1998; Porra, 2001, Swanson 1994). Second, inquiring systems have become an integral part of information systems literature and as such their use eliminates the need to use ideas from reference disciplines (Richardson and Courtney, 2004). Finally, this article 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 the discussion for the creation of the framework. The framework will map the inquiring systems on basic ISD activities so that the framework will be robust to be used in conjunction with different methodologies. In the second part the article 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 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 de-
velop 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 in 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 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:** 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 accommo-
dates 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, Croasdell, and Paradice, 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 (Shoemaker, 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/s produced is/are 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 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 being 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 in 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 method-
ologies let the information system emerge from the debate between developers and users letting multiple truths 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 is not a characteristic of the problem but of the observer: it is 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 context in which the individual is. From the individual perspective it is therefore – in principle – acceptable to use different inquiring systems according to the context rather than according to an externally defined characteristic of the problem. Individual working in isolation might tend to relay mostly on rationalism and idealism to move forward. Members of the same group will rely mostly on empiricism since members of one 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 instead 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 therefore this flexibility is not necessarily always used.

The following part of the article 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 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 15704; see also the references to ISD methodologies below): (1) Identification and Concept; (2) Requirement Definition; (3) System Design; (4) Implementation; (5) Testing and Operation.

These activities can be performed in various sequences, recursions, revisions, be omitted, focusing 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 activity 2 is in fact a fast cycle through activities 2 to 5 (Martin, 1991; Beath and Orlikowski, 1994), agile methodology if attention is given to collaboration, speed, and cyclicality (Beck, 1999; Lindstrom and Jeffries, 2004). A methodology like Multiview includes activities 1 and 2 but with a major techno-social scope (Avison et al. 1998) whereas Multiview2 (ibid) includes all activities specifically including organizational and sociotechnical analysis in activity 2. Therefore in this article the five ISD activities are considered as building blocks of ISD methodologies. Figure 1 depicts the idea of activities as building blocks. On the left are shown the ISD activities and on the horizontal axis the time dimension. The solid boxes represent the unfolding over time of a project using an agile methodology while the dotted boxes represent a process using a waterfall methodology. As figure 1 shows the activities remain the same but their sequence, specific content, and duration change.
Interpreting these five activities in terms of systems for creating knowledge, it is possible to link them to the use of one or multiple inquiring systems. The following part maps the inquiring systems on the ISD activities taking in consideration the goal of the activity, its progress, the active roles but also 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”⁵. The users discuss the situation and how they expect an IS to help them improve it. Users are usually not expert developers and therefore they can conceive some functionalities for the IS 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: users 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 what the IS has to do for the users (Sommerville and Sawyer, 1997). Functional requirements emerge from a discourse between users and developers with the purpose of reaching an agreement on the goals of the project. In this phase the groups have the possibility to behave as Singerian inquirers. The two groups debate ideas about problems and possibilities. They will tend to include into the discussion elements from all interconnected systems and they will come to an idea for what the information system has to do. When using this inquiring system, ideas are not necessarily shared by every individual. This is known and therefore 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 requirement 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 into 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.

In this situation the development group can use the Lockean inquiry with elements of the Kantian inquiry. The group is open to the environment but creates the truth, the knowledge of how the new
system has to operate, within the group. At times the group enters the Kantian inquiry when they engage in discussions about models and their fit to solve the users’ problem. As before, the members seek consensus within the group that acts as a guarantor for the created knowledge.

**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.

In the basic case 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 coded into the software. The reason for behaving as a rational enquirer resides in the need of the developers to have brief periods of “peace” where requirements can be turned into code without changes. This view has been challenged from the proponents of agile methodologies (e.g. Beck, 1999; Lindstrom and Jeffries, 2004) that propose to have the client on location during coding and paired programming (two programmers working together on the same piece of code). If these practices are carried out, then there is the possibility to use the Lockean and the Singerian enquirers. These practices seem to bring quantifiable advantages (Williams et al., 2000; Abrahamsson; 2003).

Occasionally the developer faces uncertainty regarding how well a particular algorithm fits the users’ system. If multiple algorithms have to be compared the developer enters a Kantian mode of inquiry trying to find which algorithm best fits 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 upon with the users but are the arena for criticisms, 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 code, in the 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: the 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. ISD activities are placed on the vertical axis and inquiring systems on the horizontal axis. This disposition gives origin to a grid where at each intersection an activity can be mapped on an inquiring system. In table 1 are shown the intersections that are desirable to be active during the activity according to the discussion presented above. While from case to case the grid might change the discussion above follows established and theoretically recognized relations. Ideally table 1 might be thought to show the best relationships between activities and inquiring systems but that is not the case. There
is no objective measure to quantify the goodness of fit between these two groups and the best fit depends on the specific case and on the specific attitudes of the participants.

The discussion above shows however that the situation and the physical constraints of an activity provide opportunities and limitations to the selection of an inquiring system. Therefore table 1 emerges considering both the goals but also the physical constraints of an activity. It shows that for each activity few inquiring systems are active and that throughout a project (see for example the two processes in figure 1) all inquiring systems are likely to be used and the variety and differences require attention and management. Table 1 shows that even though for complex cases a methodology based strongly on the Singerian inquiring system might be more effective (Richardson et al. 2001), not for all activities the debate required by this inquiring system is possible or desirable and therefore variety is likely to emerge.

Finally, as the case presented below will show, the framework will indicate possible consequences of the inquiring systems adopted and therefore it can be useful as a management tool.
Systems Development and Inquiring Systems

Inquiring Systems active during ISD activities

<table>
<thead>
<tr>
<th>Inquiring Systems → ISD Activities</th>
<th>Rationalism (Leibnitz)</th>
<th>Empiricism (Locke)</th>
<th>Idealism (Kant)</th>
<th>Pragmatism, Dialectic (Hegel, Singer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification and Concept</td>
<td>Activity carried out by the information systems stakeholders in the user’s organization.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Definition</td>
<td></td>
<td></td>
<td></td>
<td>Activity carried out by users and developers. Achievement of agreement cannot be postulated. Groups and individuals might bring home different views of the truth.</td>
</tr>
<tr>
<td>Design</td>
<td>Activity carried out by the development team to agree about design.</td>
<td>Activity carried out by the development team to agree about models.</td>
<td></td>
<td>Activity carried out by both teams.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Activity carried out by the individual developer alone. Fact nets are created based on internal logic. New knowledge is created outside the control of other stakeholders.</td>
<td>Activity carried out by programmers working in pairs. The developers help each other and agree on the methods used.</td>
<td>Activity carried out if during coding the client is on location. When in doubt the developer can include the client in the discussion to include his point of view in the solution of the problem.</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>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.</td>
<td></td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Theoretical Framework

The framework allows therefore seeing with finer level of detail – respect to the methodology level – the mechanisms of knowledge creation active in ISD. This framework can guide the manager in the selection of short term strategies for the exchange of information and knowl-
The framework can also show potential problems before they arise. For example, if rationalism is used during design – the programmer disappears to design the software – then it can be assumed that the fact nets will grow larger and will become more and more difficult to control as time goes by and size grows large which will turn into assumptions very difficult to change and resistance from the developer(s) to change. Therefore if rationalism is used during design then problems in the system structure can be foreseen.

Given the content of the framework, the second part of the paper is going to present the results of an ethnographical investigation of an ISD project. The scope is to show the applicability of the framework to manage knowledge creation in ISD. The case will 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 IS based on combinatorial optimization (CO). The objective of the ISD project was to develop software that could aid decision making in manufacturing 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 IT consulting company. Four universities were contributing with two team members each: one with supervisory role and the other with practical functions in the project. The manufacturer (SteelCo⁶) was contributing with four team members: two sponsors, an area manager and an internal project manager. The consultancy (Consult) was contributing with the project manager. Of the total 13 team members seven took active part in the development process. The seven 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 the system modeling (Drew), the person in charge of the organizational change process (Tom), and finally 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 never met before the project started. The team was to work together on system development for a period of three years.

The team members were located in different areas. Travel activities were kept to a minimum and 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 therefore two characteristics that made it interesting for being a good example for the argumentations presented in this paper. The first characteristic was that the IS 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, and the developers were working from distributed locations on tasks that were 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 article is based on a three 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 the development project closely. In this period I gathered data in form of observation, interviews, transcripts and minutes of project meetings and software versions.

Towards the end of the project I performed in-depth semi structured interviews with the team members 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 their e-mails for the development period. It was important to ask for the emails after the project because this would avoid biases in the content, workarounds, or opportunistic selection. The creation of the CD of e-mails was done under my supervision therefore assuring that the material was not tampered with. E-mails were an interesting data source because they contain information of direct communication among team members and therefore they help to support my field notes and my interviews’ content. E-mails are also probably the most informative documents because, as the team was distributed, most of the communication was electronic. E-mails give a very precise story of the project because the e-mail heading includes a time and date stamp, the sender and the recipients and information about attached files. From this information it is possible to infer who is active at a particular time and for which pur-
pose. The use of the “reply” button allows following discussion threads which can be used to understand which themes were considered most important by which project members (those who kept the thread going). Furthermore, thanks to the reply button, having the mails of 4 of the 7 main players\textsuperscript{8} warrants a 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 emails. Then for each activity I re-coded the data to look for which inquiring systems were used. Again the process was carried out three times. The intention was to find overlapping or dis-proofing evidence across the data sources and ultimately present an accurate case story. In the project I had to sign for the complete confidentiality of the data provided to me so I could not ask another coder to check for the validity of my observations and this is the main reason for repeating the two coding cycles on the three data sources. The use of three sources of data acted therefore as a surrogate to grant internal validity.

The Case Story

In the following a simplified view\textsuperscript{9} of the ISD activities that unfolded during this project are presented with special focus on the inquiring systems adopted by the developers.

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 a new integrated scheduling and control software to manage the activities of the inventory and the production lines. This phase was carried out by expert pro-
ject managers of the company without consulting the developers. The results were included in a project description brief that was later used as starting point of the requirement 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.

Requirements Definition

This activity was mostly done by the developers alone. The developers were taken around the factory one afternoon and then were stationed for 2 weeks at the users workstations to observe their work. Based on these two observations the developers were required to create a requirement document. Only 2 of the three developers worked on the requirement document and they did so individually. The discussion about the requirements took place in 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 perspective:

*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* (Walter).

In hindsight the other developer – Joshua – recognized this situation as not satisfactory; though his comment reflects a firm anchoring in the rational way of acting:

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

The requirement document was presented to the users in a meeting where there was no debate and the developers were left 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 that 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 (Rob).*

**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 into modules; they decided about processes for material handling; they decided about use of overtime. All 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 – now Walter had begun to actively participate in the project – 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 eventual comments would emerge during the testing of the systems modules. The project manager – Ted – once he had made the plans and defined the interfaces between modules (that were never used), stopped being actively involved in the project showing himself to be a firm believer in the rational mode of carrying out projects. This behavior is best shown in the profile of the emails exchanged during the period January 2000 to December 2001 (figure 2). The lines in figure 2
represent the emails sent by each team member over the period. To make the graph easier to read the emails of Joshua, Walter and Drew have been grouped.

![Mail Exchange between January 2000 and December 2001](image)

**Figure 2.** Emails exchanged during the LAGER project (2000 and 2001)

The emails show clearly 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 emails 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 could not completely 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 [Joshua].*
In conclusion the design phase was characterized by a rationalistic approach both for the developers and the project manager. The discussions that seemed to bring agreement did not bear any knowledge exchange and were therefore not transformed 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 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 could have been discussed later during the testing phase. In this phase the developers also suffered from the project manager decision to step back though 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 [Joshua, interview]*

The developers also used idealism as a mode of inquiry to a large extent. E.g. 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 best fitting model (usually a statistical distribution). Similarly to the other simplifications, the developers thought that the users would validate the use
of these models during testing. During coding the developers behaved therefore as expected in the framework (table 1).

**Testing and Operation**

During the development project the development team held three demonstrations of the software on the 16th, 19th, and 21st month 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 led firmly the discussion impeding the users the possibility to criticize the validity of the simplifications in the software. These testing phases were not done accordingly to the Singerian inquiring system seeking ardent debate. There was very limited knowledge creation that would have brought the project closer to being successful.

This project joined the other 84% of ISD projects that fail each year\(^{10}\). 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 pointy comment about the developers’ behavior that indicated precisely where the root of the problem was:
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 [Joshua].

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 emarginated immediately. The following table 2 shows the overview of the inquiring systems active for the developers during the different activities of the project. In block letters are reported the DESIRED inquiring systems from table 1.
Inquiring Systems active during LAGER ISD activities for the developers

<table>
<thead>
<tr>
<th>Inquiring Systems → ISD Activities</th>
<th>Rationalism</th>
<th>Empiricism</th>
<th>Idealism</th>
<th>Pragmatism and Dialectic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification and Concept</td>
<td>DESIRED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(not treated in the case)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Activity carried out by developers singly. Users passive.</td>
<td>DESIRED</td>
<td>DESIRED</td>
<td>DESIRED</td>
</tr>
<tr>
<td>Design</td>
<td>Activity carried out by developers singly. No discussion about coordination in the initial phases of the project.</td>
<td>DESIRED</td>
<td>DESIRED</td>
<td>DESIRED</td>
</tr>
<tr>
<td>Implementation</td>
<td>DESIRED</td>
<td>Activity carried out by the individual developer. New knowledge is created outside the control of other stakeholders.</td>
<td>DESIRED</td>
<td>DESIRED</td>
</tr>
<tr>
<td>Testing and Operation</td>
<td>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 versa.</td>
<td>DESIRED</td>
<td></td>
<td>DESIRED</td>
</tr>
</tbody>
</table>

Table 2. Practice Framework for the Developers

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 however the result of the predilection of the developers for the rational inquiring system rather then the result of a purposeful decision.

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, distributed team members etc. – the case shows that choosing a specific path of knowledge creation brought consequences to how the project unfolded. The long time used to create the software resulted in very complex fact nets that were very difficult to change even for the programmers that made the nets, this influenced the programmers’
acceptance of users’ comments – or lack of acceptance – and ultimately shaped the content and capability of the finished product. Both the developers and the clients remained dissatisfied with the way in which knowledge was exchanged during the project. This dissatisfaction is not value free since monitoring how knowledge was exchanged during the project could have allowed the project managers to take specific measures – not least to kill the project and save money. The analysis can be continued at two levels: activity and global level.

At the activity level the analysis shows that the developers followed mainly a single mode of inquiry even where other modes would seem more adequate. This is in line with the results of Kienholz (1999) that says 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) proved that given the right preparation and context people used to rational inquiry can shift their style and work as Singerian inquirers. The laissez faire style of the project manager (Ted) not only left the developers to their own devices but also did not guide them through the delicate process of adapting their inquiring style.

At global level this study shows that, despite the rational reasons for promoting Singerian-type inquirers in all ISD activities, there is a gap between the theory and the practice and that this gap is part of normal and traditional ISD practices and therefore very difficult to close. One way of proceeding towards 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, Courtney and Paradice, 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 are serving 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. The main implication of the analysis presented above is showing the connection between the disconnected behavior of the developers and the fact nets. Developers will tend to develop the software partially from the discussion with the users and partially from their own reasoning. If this integration results in large fact nets then the discovery of all the causal relationships that have been embedded in the nets is almost impossible. Furthermore the complexity of the fact nets influences negatively the users’ understanding of the code and therefore decreases the power of the users to argue for changes. Finally the investment made in the work done by the developers is a natural source of resistance to users’ feedback. On the contrary if the knowledge integration results in small fact nets they can be easily analyzed and invalidated during testing. Time intervals between testing activities should therefore 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.
Cyclicalinity. In the identification and in the design activity consensus is achieved in different
groups. Therefore there is no a priori guarantee that the system will serve the users correctly.
However the more often the Singerian inquiring system is used the more likely it is that the
resulting information system would serve all involved interests. The direct consequence of
the identification of the fact nets in the code is in showing the importance of going through
multiple cycles of designing, implementing, and testing. In process structuring this translates
in planning to go through the testing activity many times putting developers and users in the
condition of debating the work done according to the Singerian inquirer. Short time windows
and small tasks are advocated above are the key to making this iterative process possible.

Managerial implications

Support the Singerian Inquirer. Entering in the Singerian inquiring mode is not easy. A pres-
entation, 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,
Courtney and Paradice 2001). Project managers on both sides should agree to mediate the dis-
cussion instead of leading it and should structure it so that all necessary elements are swept in.
Managers are also the guarantors that ethics is 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 blackboxed, in other words developers can-
not attack with critical spirit their own fact nets because these have emerged from the develop-
ers 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 manag-
ers 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 provide the connection between the activity and the methodological level. The guidelines do not reject existing methodological practices but rather add a new knowledge driven angle to them. For waterfall-type developments (Boehm, 1988) the takeaway would be to speed up development and to identify and develop working modules that can be put into operation within short delays in this way knowledge can be collected from actual use. For spiral-types developments (Boehm, 1988) the takeaway would be to go through the cycle rapidly before the fact nets become too big. 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 (Lindstrom and Jeffries, 2004).

Conclusions

This paper presents a view of ISD where the software creating activities and managerial activities are joined by knowledge producing activities. The article shows that it is possible and advantageous to move away from an overreaching philosophical base for an ISD methodology and to map the basic ISD activities on the inquiring systems based on different philosophies. The article 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 therefore diversified ways of creating knowledge which in turn have to be supported by multiple and adequate practices.

A case study is presented where it is shown that - while it is not possible to quantify the fit between ISD activities and inquiring systems – the framework can be used to predict the consequences of the chosen path.

Based on the difference between the theoretical framework and its specific instance from the case study it is concluded that key elements for ISD success can be found in a conscious focus on the knowledge dimension of ISD and in particular on: speed and cyclicality, size of modules, and type of testing situation.

References


---

**Notes**

1 Low-level here is not a diminishing term. It refers to the activities like design, coding, testing as opposed to the “high level” which is the sequence in which these activities unfold: the process.

2 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

3 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.

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

5 The word “users” here is a simplification of Churchman’s (1971 p. 43) notion of “client” as one of the main roles in system design. While the users might have multiple interests, for the scope of this article a monolithic view of users is sufficient because with respect to the IS developers they would look like a compact community.
The names of the companies as well as those of the project participants have been disguised.

I interviewed all but two team members. One team member had changed job. The second abandoned the project only 6 months in the development.

As said in note 4, one team member had changed job. The other abandoned the project very early in the development. The third was using an email system not accessible from outside his organization.

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 here. This section aims to present few instances of the activities discussed above to show that also in real cases multiple inquiring systems are active.