The Discursive Constitution of Software Development

Francis Cornut
Department of Management
London School of Economics and Political Science

Thesis submitted for the degree of Doctor in Philosophy
February 2009

Abstract

The successful development of software continues to be of central interest, both as an academic topic and in professional practice. Consequently, several software development approaches and methodologies have been developed and promoted over the past decades. However, despite the attention given to the subject and the methodical support available, software development and how it should be practiced continue to be controversial.

This thesis examines how beliefs about software development come to be socially established as legitimate, and how they come to constitute software development practices in an organization. It is argued that the emergence of a dominant way of conceiving of and practicing software development is the outcome of power relations that permeate the discursive practices of organizational actors. The theoretical framework of this study is guided by Pierre Bourdieu's theory of symbolic violence and organizational discourse theory.

As a research method, ethnographic research techniques are utilized as part of a case study to gain deep insights into the standardization of software development practices. The research site is the IT division of a large financial services organization and is composed of ten units distributed across eight countries. The tumultuous development of a knowledge management programme intended to institutionalize a standard software development process across the organization's units provides the case for this research.

This thesis answers the call for studies providing detailed accounts of the socio-political process by which technically oriented practices are transferred and standardized within organizations. It is submitted that a discourse theoretical approach informed by Bourdieu's thinking enables us to conceptualize this process in a more meaningful, and theoretically rigorous, manner. In providing this theoretical approach, the thesis seeks to contribute to current research on technology and innovation management, and to offer guidance on some issues concerning the management of the software development process.

Acknowledgements

Throughout the period of this PhD, my supervisor, Professor Chrisanthi Avgerou, has generously given of her time and wisdom. In supporting me on this intellectual journey, Chrisanthi has helped me become more the person I want to be. I simply could not wish for a better supervisor and mentor. First and foremost, my thanks go to her.

I am also indebted to Dr. Shirin Madon who, while serving as PhD Programme Director, lent me an attentive ear and helped me overcome the obstacles I encountered. Shirin has had a positive impact on my life – probably a lot more than she might imagine. I can never hope to repay her kindness.

Information Systems and Innovation Group at the London School of Economics is a vibrant intellectual environment, and I am indebted to its members who have contributed to my development on a day-to-day basis. In particular, I would like to thank Dr. Edgar Whitley who taught me many of the things that a researcher must know.

The LSE Studentship awarded by Information Systems and Innovation Group provided substantial resources for this study. I am grateful for its generous support.

Finally, on a different note, I wish to thank members of the case study organization with whom I worked for nearly a year. Special thanks to Alex, Corinne, Francis, and Richard.

Contents

List of Tables

List of Figures

1 Introduction

Four decades ago, it became recognized that developing good software was difficult. This emerging reality was in evidence in the high proportion of projects completed late and over budget, and in the density of errors in systems. Too often, systems did not function as intended and had to be substantially redesigned. The causes of what became known as the "software crisis" were multiple and cumulative, but essentially stemmed from general inexperience in developing large and complex software.

The software crisis affected both the corporate world and the military, but it is from the latter quarter that a response was first heard. In 1968, the NATO Science Committee convened a group of computer scientists and captains of industry to plot a course out of the software crisis. The general consensus that emerged from the NATO conference was that the answer to the software crisis lay in changing software construction from an ad hoc skill to an engineering discipline. In practical terms, this entailed formalizing software development.

There are two fundamental ways in which an organization can go about formalizing the development process. First, an organization can acquire a methodology and adapt it for its own particular needs. A methodology generally consists of a set of guidelines, techniques and tools based on a particular philosophy of software development (Avison and Fitzgerald, 2003; Wynekoop and Russo, 1997). A methodology makes explicit the tasks to be completed and restricts the number of arbitrary ways in which they can be completed. Secondly, an organization can implement a software process improvement (SPI) scheme to manage and control its software development process. SPI involves assessing the quality of the process and finding ways of systematically improving it (Paulk et al., 1993). Today, it is the capability maturity model (CMM) that crystallizes the basic concepts of SPI; as a result, CMM is widely regarded as the reference model in the field.

Methodologies and SPI schemes are associated with the engineering approach to software development because they encapsulate the scientific principles known to established engineering disciplines. While there has been a marked tendency to

attempt to resolve the difficulties associated with developing software by relying on formal practices based on scientific engineering principles, there has always been a feeling of uneasiness with such practices. For as long as software development has been practised, it has been recognized that this activity requires a great deal of intuition and creativity. There is a sentiment that formal practices impose unwanted restrictions on the software development process, and that formal practices may take away the human aspect that is required in software development.

It is perhaps the apparent irrationality of software professionals and their practices that has, more than anything else, provided an impetus for the development and diffusion of formal practices. For outsiders to the field- particularly observers adopting the instrumental rationality of engineering- software professionals often appear to engage in activities that have little value. Instrumental thinking suggests that when an activity is pursued rationally – systematically, with a focus on the desired results – difficulties will be minimal. According to this line of reasoning, the problem of completing projects on time and within budget is principally attributable to the irrationality of the practices on which individuals are reliant (Baskerville et al., 1992; Introna, 1996).

Critics have argued that software development practices are often the expression of a "hidden rationality," reflecting a need to acquire a "complex and disparate view" of the development process (Stolterman, 1991). It is suggested that many situational and contextual factors may influence a developer's decision to pay attention to one thing, rather than another, at a given point in the development process. This decision may not always appear rational in a strictly scientific sense; nonetheless, many software professionals believe it to be justified. For example, Naur famously argued that "software development in all its phases, and irrespective of the techniques employed in its pursuit, must and will always depend on intuition." (Naur, 1985). If it is to be believed that software development normally necessitates intuition and tacit knowledge, there is a strong argument for promoting the value of the practical and situated rationality of humans and discouraging the kind of universal rationality that methodologies and standardized SPI approaches promote.

In a similar vein, it is a frequently repeated statement that the universal laws on which traditional engineering disciplines are based do not necessarily apply to software development. Several commentators have pointed out that unlike the construction of a bridge or an airplane, during which projects, the laws of physics remain the same, the construction of software cannot be pursed as a strictly-sequenced activity—and, certainly, not as an automated one. In practice, many details only become known to software professionals as they progress in the implementation. Even if all the relevant facts are available to them at the outset of a project, human beings appear unable to comprehend fully the plethora of details that must be taken into account to design and build software (Parnas and Clements, 1986; Simon, 1957). This fact has led some commentators to conclude that software development has little to do with engineering, and that imposing the formalism of traditional engineering disciplines on software professionals will not improve the efficiency of the process or the quality of the applications.

Today, there may or may not be a software crisis, but there is definitely what might be called an "identity crisis" (Keil-Slawik, 1996). This condition is evidenced by the fact that no consensus as to what software development is, and how it should be practiced, has been reached after more than four decades, since the software crisis was first identified. To be sure, considerable advances have been made in the design and construction of software, and in the management of projects. Software projects are overall better managed than they used to be and programmers, more productive than ever (Krishnan et al., 2000; Sauer and Cuthbertson, 2003). However, the fact remains that software professionals are, even today, persistently engaged in an ideological struggle to justify their beliefs and their use of particular practices. The phrases that commentators use to describe this struggle, such as "emotionally complicated topics" and "methods wars" (Boehm and Turner, 2004), appear very apt.

It is important to recognize that the application of methodologies and SPI schemes in practice and in attitude is never an all or nothing proposition. Past research suggests that software engineering standards regarding practices tend to be adapted and used on a piecemeal basis, rather than being consumed in entirety or completely rejected (Russo and Stolterman, 2000a). In this sense, software professionals appear to form an appropriate software development approach out of the intellectual and material

resources (i.e. ideas and artefacts) available to them. Moreover, it would seem that such a process is informed by what the developer conceives software development to be (Russo and Stolterman, 2000b). This research endeavours to unpack how a collective agreement about the nature of software development may be negotiated, and what it should involve in terms of practices.

1.1 The standardization of software development

This thesis reports on an in-depth, qualitative case study documenting the tumultuous development of a knowledge management program intended to institute a standard software development process within an organization. The study organization is the IT division of a large investment bank and is composed of ten units, or software houses, geographically distributed in ten locations. The knowledge management program was primarily motivated by the perceived need to improve the quality of the software products and the efficiency of development process.

For the purpose of synergy and efficiency, organizations often engage in cross-unit transfers of internal best practices. Internal best practices are those work practices that are believed to be more effective at delivering particular outcomes than any other internal practices. In the past decade, the transfer of internal best practices has been high on the priority list of most IT organizations, and has been seen as critical to a firm's ability to build a competitive advantage (Dyer and Singh, 1998; Szulanski, 1996). Consequently, a large number of IT organizations have undertaken programs with the objective of becoming able to systematically assess their practices and to ensure that the practices that are proven superior are institutionalized within the unit in which they are found or across the organization.

Prior to the commencement of the study, the ten software houses had established their own software process based on their respective best practices. The knowledge management program initiated aimed at constructing a standard organizational software process from the software houses' best practices. The program entertained a vision of an organization in which software development would be done in a standardized way across geographical boundaries, and in which practices found superior in one unit would be seamlessly transferred across the whole organization.

This vision had earned the knowledge management program the name 'learning organization' program.

The learning organization program resonated with several ideas that had pervaded the management discourse for the past decades. The most influential of such ideas included the application of management approaches aimed at creating process-oriented organizations and fostering continuous and effective learning (Hammer and Champy, 1993; Senge, 1990). Like countless other corporations, the study organization had found in those popular ideas an approach to determine how to best construct its work processes and to improve the quality of its output. In this sense, the learning organization program seemed, at first sight, rather innocuous; it appeared to be just another attempt to do more with less. However, as this thesis will gradually demonstrate, the program had far-reaching consequences for the organization.

Commentators have noted that the sort of program initiated by the study organization entails an intervention in the organizational culture and in the practices of software professionals. For Ngwenyama & Nielsen (2003), for example, it "is an attempt to change how software professionals think and act in their everyday organizational activity." In this study, the learning organization program is understood as: an attempt to change how a majority of the software professionals think about and practice software development.

Commentators have observed that the implementation of programs intended to improve the development process and the quality of software applications often turns out to be a long and complex process because organizational actors find themselves confronted with dilemmas based on contrary demands and value conflicts (Iversen and Mathiassen, 2003). In the study organization, these dilemmas were in evidence in the difficulties associated with reconciling the needs for predictability and efficiency with the need for adaptability and creativity. Consequently, devising a standard software process involved negotiating the degree of control and adaptability that software development should allow. Finding this balance proved to be arduous because of the dissimilarity in ways of conceiving and instantiating software developed in one software house, as opposed to another.

For the software houses involved in this study, adhering to a standard organizational process entailed abandoning valuable knowledge that comprises a tacit component. Face-to-face interactions and work within small groups, within a given software house, had produced highly idiosyncratic practices. Within the software houses, the development process was built from knowledge embedded partly in individual skills and partly in collaborative social arrangements. This study suggests that the constitution of standard organizational software process is a socio-political process involving negotiations among organizational actors to determine how software development should be thought of and practiced.

The organization had set itself the objective of institutionalizing a standard software process a year before the research began. During that year and the ten-month period during which fieldwork was conducted by the researcher, the organization experimented with different ways to come up with a standard software process. It initially attempted to develop the standard process from its own best practices, and failing this, went on to make use of a well-known methodology said to embed industry best practices. Both attempts turned out to be plagued by major difficulties. Over a period of almost two years, the organization was exposed to many ideas and practices. Some appeared highly appealing and exciting, while others disappointed and caused anxiety. This study unpacks how different visions of software development were constituted over almost two years, and how the values of the practices these visions presuppose were negotiated.

The research approach is designed to achieve an in-depth understanding of how individuals belonging to the same organization go about negotiating the legitimacy of different ways of conceiving of and instantiating software development. This study pays particular attention to the ideas that pervade the context and that influence individuals. A key idea put forward in this study is that software development is the outcome of the communicative practices of software professionals. Software development is here seen as being constituted through the practice of writing and speaking; that is to say, through the production, diffusion, and interpretation of written and spoken texts (Oswick et al., 2000). Thus far, scholars have tended to see software development as being shaped by systems development methodologies and

methods (Avison and Taylor, 1997; Boehm and Turner, 2004; Russo and Stolterman, 2000a), or by some philosophical and paradigmatic underpinnings found in the research literature (Constantine, 1993; Hirschheim et al., 1996). The focus of this study is in contrast to the practical situated context in which development takes place. It is submitted that approaching software development as being locally discursively constituted enables us to conceptualize in a more meaningful, and theoretically rigorous, manner the process by which beliefs about software development and practices are established as legitimate. In providing this theoretical approach, the thesis seeks to contribute to current research on software development.

An important proportion of the total budget of the learning organization program was allocated to fund SPI-related activities. This study focuses on SPI practices for two reasons. First, it is through SPI that the legitimacy of development practices was negotiated in the study organization. SPI provided the mechanism for negotiating the value of ideas across many locations and served as a means to constitute a standard software process. Moreover, the SPI practices in place, themselves, reveal a lot about what software development is in a particular organizational context. As will be explained in the details of this thesis, an organization that values efficiency and predictability is more likely to invest resources in the management of software development process than an organization that does not.

In light of the debate over the nature of software development, the overall research question addressed in this study is: **How do beliefs about software development and software development practices come to be socially established as legitimate in a software development organization?** This general research question is further defined in the following way: How are the contradictory needs for creativity and agility, on the one hand, and efficiency and productivity, on the other, negotiated? What bearings does the institutional context have on the manner in which software development is principally thought of and practiced in a software organization?

1.2 Contributions of the research

Researchers have identified a bias towards normative studies and surveys to account for what is happening in software organizations. As such, studies generally seek to

provide guidance for the selection and implementation of methodologies, tools, and approaches without paying much attention to organizational and social elements (Hirschheim and Klein, 2000; Wynekoop and Russo, 1997). As Russo & Stolterman (2000a) note, "Only a small percentage of information systems methods has employed in-depth studies of the actual process of system design and development". Consequently, there is a great need for "in-depth studies of practice" that "create rich descriptions of practice, and come up with interpretations and analysis of this practice."

This study answers the call of IS researchers for studies providing detailed accounts of the practices of software professionals. To obtain such an account, an in-depth immersion in the field was necessary. The focus is placed primarily on the practices of text production, diffusion, and interpretation of software professionals. The practice perspective chosen for this study has not been widely used in the IS field, and this study will yield some useful methodological findings. The adoption of a practice perspective informed by discourse theory will help to capture the influence of the institutional context within which individuals are situated, while acknowledging the purposeful, result-oriented nature of their practices.

The research conceptualizes software development by devising a theoretical framework from theories which are known within the social sciences, including organization studies, but which are, so far, unfamiliar to IS research. Pierre Bourdieu's theory of symbolic power and Norman Fairclough's critical discourse analysis are joined together to probe how software development is constituted, and why it is primarily constituted in a certain way, rather than in others, in a particular development context. Previous research has recognized that software development can be conceived of and instantiated in different ways (e.g. Constantine, 1993; Hirschheim et al., 1996; Iivary et al., 1998), but this constitutive process has remained unexplained (for a notable exception see Madsen et al., 2006). Moreover, because of the predominant normative outlook they have adopted, IS researchers have been generally preoccupied with finding the conditions that render certain forms of software development successful, without paying much attention to the organizational and social factors that cause their emergence and acceptance at grassroots level (e.g. Boehm and Turner, 2004). The theoretical framework was

devised with the aim of filling in this important gap in our understanding of software development.

1.3 Structure of the thesis

In this introductory chapter, issues surrounding the practice of software development have been introduced. The constitution of software development has been identified as the topic of the thesis. The role of tools, methods, and methodologies has been drawn to the foreground. Following this, a highlight of the contributions has been provided. The remaining of the thesis is organized as follows:

Chapter 2 describes the different ways in which software development has been conceived since the software crisis was first identified. An historical account suggests that tools, methods, and methodologies have played a centrally-important role in defining, at particular points in time, how to rationally practice and conceive software development. It is also in this chapter that the research questions are developed.

Chapter 3 establishes the research framework. The theoretical framework takes as its point of departure organizational discourse theory. Critical discourse analysis and the theory of symbolic power are combined to form a theoretical framework applicable to developing an enhanced understanding of the organizational phenomenon under study. The roles of institutions and the theme of power are central to the framework. The theoretical framework is sensitive to the effects of unequal power relations among organizational actors, and to the influence of institutional forces.

Chapter 4 describes the important contribution that methodological choices have made to this research with regard to the constitution of software development. It is argued that the two main paradigms in social science research, positivism and interpretivism, have, in the present case, substantial limitations which can be bypassed by relying on a practice perspective. The case study is identified as an appropriate research strategy. The application of Paul Ricoeur's hermeneutics of suspicion as an epistemology of data interpretation is justified.

Chapter 5 presents the research setting and the knowledge management program intended to institute a standard software development within the organization. IBTech is an IT organization responsible for developing and maintaining the software products of an investment bank. A key feature of the organization is that its members are geographically-dispersed. The knowledge management program, whose evolution was followed over a 10-month period by the researcher, is principally based on the precepts of the CMM.

Chapter 6 presents the data collected and analyzed. Analysis of the organizational texts collected has revealed that seven organizational discourses form two conflicting ways to thinking of and practicing software development. Significantly, those two visions possess different degrees of legitimacy within the organization.

Chapter 7 theorizes the findings by critically engaging with Fairclough's critical discourse analysis and Bourdieu's theory of symbolic power. The analysis reveals how the particularities of the socio-institutional context induce organizational actors to practice and discursively represent software development the way they do. The correspondence between the micro and the macro is illuminated by paying particular attention to the relationship between the discursive and non-discursive practices of organizational actors, on the one hand, and the influence of the socio-institutional context, on the other.

Chapter 8 provides an overview of the thesis as a whole. The core contributions that this research makes to the IS literature are presented and followed by a discussion of the limitations of the research. Finally, some key areas for future research are identified.

2 Literature Review

Software development has been thought of and practiced in different ways over the past four decades. This chapter begins by chronologically tracing the evolution of software development. This evolutionary sketch suggests that tools and methods have played a significant role in defining how software development should be practiced in different contexts. The chapter then examines how the changing nature of software development can be accounted for. The research questions are also developed in this chapter.

2.1 Past debates and established ideas

2.1.1 Software development as applied science

Until the 1960s, very few individuals were professional programmers. Programming was essentially an ad hoc skill that technical individuals developed to resolve the problems of their fields involving the use of computers (Cerruzi, 1998). Consequently, there were no established principles or standards for writing code methodically. This unsystematic approach caused no major problems as long as programs remained simple.

As transistors replaced vacuum tubes in the early 1960s, computer performance improved spectacularly. More complex software projects were initiated and the problem of software started to dominate the problem of hardware. The lesson that was being learned at that time, simultaneously in many projects, was that software projects which were large, complicated, and involved unfamiliar aspects were particularly vulnerable to large, unanticipated problems. In the same vein, it was also becoming increasingly clear to software professionals that software programming did not scale up linearly; big projects required proportionally more manpower than small projects (Brooks, 1975).

The ‘software crisis’ was a term introduced in the late 1960s to describe the impact of rapid increases in computer power and the complexity of the problems which

could be tackled. The software crisis manifested itself in projects running over-budget and over-time, software of low quality, and code difficult to maintain. Some authors also identified programmer shortages as central to the crisis. The software crisis affected both the military and the corporate world, but it was particularly evident in America, where more than half of the global stock general-purpose computers were (Campbell-Kelly, 2003: 90).

The NATO Science Committee sponsored the first conference on software engineering in 1968, an event that was widely regarded as marking the origin of the software engineering discipline (Friedman, 1989; Mahoney, 2004). The conference set out to plot a course out of the software crisis. In defining software engineering as "the application of systematic, disciplined, quantifiable approaches to the development, operation, and maintenance of software" (NATO Science Committee, 1968), the committee made clear that the answer to the software crisis lay in changing software programming from an ad hoc skill, to an engineering discipline. In particular, it was recognized that mathematics would have to play an appropriate role in software engineering, just as it did in well- established engineering fields. Whether software development lent itself to the application of a scientific approach was not questioned.

During the late 1960s and early 1970s, the conviction that the use of sound programming technique, connected with mathematics and computational science, could improve the quality of the software gradually solidified. Formal methods were to provide the mathematically-based techniques for the construction of systems, especially large scale systems (Banach, 2007). In effect, it was believed that formal methods, such as structured programming, could offer a way out of the crisis.

Dijkstra coined the term "structured programming" in 1969, in a now-classic article that emphasized the importance of error prevention, as opposed to error cure (Dijkstra, 1969). Briefly summarized, structured programming is a programming paradigm that limits the number of arbitrary ways in which a program can be written (see also Jackson, 1975). This paradigm is most famous for removing or reducing reliance on the 'goto' statement, a statement criticized for producing code that is unreadable and, generally, not maintainable (Dijkstra, 1968; Knuth, 1974).

Structured programming is based on a top-down approach. It recommends breaking down a program into simple subroutines with a single point of entry and using local variable within such subroutines. As a consequence, this programming paradigm makes it easier for programmers to understand small pieces of code without having to understand the whole program at once. Furthermore, it makes it easier to implement and test small operations, and tie them together into the whole program (Jackson, 1983). Dijkstra successfully made structured programming the educational standard, but did not succeed in making it a strict requirement.

In sum, during the late 1960s and early 1970s, the conviction that programming techniques could improve the quality of software prevailed. This conviction resonated with the sentiment that the software crisis could be hit at its core by establishing a strong mathematical foundation. The argument was put forward that expressing program designs in simple algebraic forms made it easier to avoid serious mistakes, and made the complexity of programs manageable. The sense of comfort that structured programming and other such programming techniques had created did not last long, however.

Boehm demonstrated in 1973 that the majority of errors in software were made during the design phase-- that is, before the construction of the software begins (Boehm, 1973). Moreover, being a mathematician, Boehm could make his point using the legitimate language of the time, mathematics. The argument laid out suggested that a solution to the software crisis was not to be found in better programming techniques. But Boehm did not stop there; three years later, he delivered a strong criticism of the mathematical approach to software engineering:

> Those scientific principles available to support software engineering address problems in an area we shall call Area 1: detailed design and coding of systems software by experts in a relatively economics-independent context. Unfortunately, the most pressing software development problems are in an area we shall call Area 2: requirements analysis design, test, and maintenance of applications software by technicians in an economics-driven context. (Boehm, 1976: 67)

In defining software engineering as "the application of systematic, disciplined, quantifiable approaches to the development, operation, and maintenance of software," the NATO Scientific Committee had omitted the term, 'design.' Being predominantly scientists and practitioners of a mathematical bent (physicists, engineers, etc.), members of the committee did not see design as being relevant, as it was felt to be outside the domain of the 'measurable.' However, by defining software engineering as "the practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them," Boehm (1976) sought to give design central importance.

More significantly, the work of Boehm initiates a shift in the way software development should be conceived. The US government, in general, and the Department of Defense, in particular, were important consumers of programming services, as well as sponsors of several of the large software projects. Many of these projects were conducted under the pressure of the competition between the USA and the USSR during the Cold War, and, more specifically, during the space race. The fact that NATO sponsored the first conference on software engineering is evidence of the strong presence of the military in the software world. However, governments are generally less subject to economic constraints than businesses, especially in a militarily-tense climate where national security is believed to be under threat. Thus, Boehm sought to develop a vision of software development that is more in tune with the logic of the market than with the logic of the military.

2.1.2 The methodology movement

The work of Barry Boehm contributed to turning attention to analysis and design practices. The creation and diffusion of structured analysis and design methodologies quickly followed, giving rise to the methodology movement. Methodology set down the necessary steps to be taken to develop software applications. These steps typically involve ascribing roles to individuals, formalizing the process of software development into a discreet number of stages, and specifying a full set of necessary and sufficient activities within each stage. Although structured analysis and design methodologies differ at the level of their techniques, they all purport to provide a

framework for mapping the behavior of the system and the ways in which users will interact with it.

Interestingly, different countries have tended to develop and use their own types and styles of methodologies (Avgerou and Cornford, 1993). Moreover, the diffusion of methodologies has occurred through different channels across countries. In the United Kingdom, Structured Systems Analysis and Design Methodology (SSADM) was produced for the Central Computer and Telecommunications Agency (CCTA), a governmental office concerned with the use of technology in government. SSADM quickly achieved significant penetration in commercial IT because the government frequently imposed it on contractors. In America, on the other hand, the means of diffusion of structured analysis and design methodologies occurred principally through books and training materials intended for practitioners. De Marco, Yourdon, and Constantine were the figureheads of the new structured analysis and design movement. In 1978, De Marco published his seminal work, "Structured Analysis and System Specification" (De Marco, 1978), which signalled the beginning of the movement's bandwagon. A year later, Yourdon and Constantine paired up to produce "System Design" (Yourdon and Constantine, 1979). To accelerate the propagation of his ideas and those of his like-minded colleagues, Yourdon established his own publishing house, Yourdon Press.

In the early 1990s, the diffusion of methodologies led the development of computer-aided software engineering (CASE) tools. CASE tools transferred the precepts and practices of methodologies into software intended for software professionals. Such tools arose out of developments such as Jackson Structured Programming and the software modelling techniques of SSADM and structured programming. CASE tools placed particular emphasis on analysis and design, but they frequently comprised modules and tools that supported code reuse, code translation, project management, and testing and quality control.

The methodologies movement had a decisive impact on software development in many respects. First, it established analysis and design as necessary steps in the development process. As a result, software development effectively became an activity concerned with the establishment of a mix of social and technical

considerations. This prompted an interest in the so-called 'hybrid manager.' Earl & Skyrme (1992) argued that if a such a role could be created, filled by an individual with both a solid understanding of software development and a knowledge of the particular business, greater success would be achieved in development. Although the welcoming of soft elements seemed, at first sight, to take software development along a totally different path than that originally cleared by the scientists of NATO, it is must be recognized that the methodology movement also contributed to bringing software development close to industrial engineering.

The development of methodologies helped to popularize the application of scientific management to software development. Within academia and industry, it is still today a frequently-repeated statement that methodologies make possible the dissemination of an understanding of software development and development practices (Avison and Fitzgerald, 2006). Although methodologies often differ considerably (Avison and Taylor, 1997), they are almost always an application of division of labour to software development: they break software development into a discrete number of stages, prescribe a full set of necessary and sufficient activities within each stage, define the roles and responsibilities according to required technical competence, and seek to impose a 'one best way' on (knowledge) workers.

It is with Taylorism and the factory system in mind that the 'software factory' approach was developed (Johnson, 1991; Swanson et al., 1991). This approach suggested that software organizations could become more efficient and flexible by relying extensively on the strategic reuse of codes, product specifications and designs, documentation and manuals, test cases, and personal experience. The adoption of the software factory approach required the deliberate (rather than accidental) sharing of resources across different projects within an organization. The approach aimed to make software organizations benefit from economies of scope, "cost reductions or productivity gains that come from developing a series of products within one firm (or facility) more efficiently than building each product from scratch in a separate project" (Cusumano, 1991: 8).

Despite the fact that the term 'software factory' quickly became part the IT buzzword lexicon, the application of mass production techniques to software development was,

in fact, nothing new. Already in the 1960s, subroutine libraries that were reusable in a broad array of engineering and scientific applications were built and used (Cusumano, 1989; Mahoney, 2004). The idea that software should be built from prefabricated components was first published in Douglas McIlroy's address at the first NATO conference on software engineering (McIlroy, 1968). However, the software factory concept received attention in part because it resonated with another approach that also received a great deal of attention in the early 1990s, object-oriented system development (OOSD).

The object-oriented approach suggests that a computer program be viewed as a collection of objects interacting with one another. This view stands in marked contrast to that of the structured approaches introduced above, which conceptualize a program as a collection of functions or procedures. For this reason, it is often claimed that OOSD brought about a shift in paradigm.[1]

According to several practitioners, the effective reuse of component makes software development practices more efficient than those previously used (Booch, 1994; Coad and Yourdon, 1991; Jacobson et al., 1995; Johnson, 2002). Until recently, however, researchers expressed serious concern about the value of this approach. Several studies suggest that the adoption of object-oriented technology is often more a matter of fashion than of reasoned development (Smith and McKenn, 1996). For example, Briand et al. (1999), who studied the impact of object-oriented on systems development practices, note that "[object-oriented] technology adoption is mostly the result of marketing forces, not scientific evidence."(Ibid.: 388)

In retrospect, the 1980s and 1990s witnessed the emergence and growth of a mixture of instruments intended to stabilize software development. In particular, this period was marked by the proliferation of methodologies and CASE tools designed to provide knowledge about the required practices of software development. At a more conceptual level, methodologies and CASE tools also provided adopting

[1] The researcher does not see object-oriented as a programming paradigm, but rather as a design paradigm. A system is designed by defining the objects that will exist in that system. The code that actually does the work is irrelevant to the object, or the people using the object, due to encapsulation. For this reason, the term "object-oriented approach" is in this text used instead of "object-oriented programming".

organizations (perhaps unintentionally) with a vision of what software development was and how it should be practiced (Andersen et al., 1990; Mathiassen, 1998). The vision in force was the software factory. Although it is industrial engineering and scientific management, rather than mathematics and computational science, that provided the roadmap for the evolution of software development, one would presume that sufficient progress would have been made to overcome the software crisis. However, the 1994 figures of an America research firm, The Standish Group, implied that the software crisis was still very much alive.

A result from the Standish Group's CHAOS report that received a considerable amount of attention was the reported 189% average cost overrun on so-called challenged projects (ie, projects not on time, on cost, and with all the specified functionality) (The Standish Group, 1994). The report has been criticized on many grounds, and the question of whether it meets the standards of a good academic study is still debated today (Jørgensen and Molokken-Ostvold, 2006). The popularity of the report among practitioners, and the rate at which it was cited in the mass media and by consulting firms, nonetheless says something important about the state of software development: a wealth of ill-founded, yet compelling, claims appear to have been formulated to maintain a climate of crisis. Consulting firms, technology vendors and other such knowledge merchants may be the most to blame, as it was they that had an interest in maintaining a sentiment of crisis in order to promote their ideas and products as crisis solutions (Glass, 2006). In a similar vein, the fact that the software crisis was still talked about, and that products and approaches were being developed and presented as 'crisis solutions' in the 1990s, suggests that the discipline remained anxious about the quality and appropriateness of its practices.

2.1.3 The quality turn and software process improvement

In the early 1980s, the concept of quality became widespread in the business world and companies began to implement quality improvement methodologies such as Six Sigma, Total Quality Management, and Zero Defects. While the principles behind modern quality management originated decades earlier with the work of Walter Deming, it is in the 1980s that quality management became explicitly articulated

with the concept of (business) process popularized by Peters and Waterman in the book, "In Search of Excellence" (Peters and Waterman, 1982).

The popularity of quality management reached the software industry and gave rise to the Capability Maturity Model (CMM). The CMM is an application of quality management and process improvement to software development. The original concept of the framework was developed in the early 1980s by Watts Humphrey at IBM. Humphrey's unique insight was that software organizations had to remove impediments to continuous improvement in a specific order if they wished to keep improving their processes capability over time. Appendix 2 describes the components of the framework.

The development of the CMM began formally in 1986 through a collaboration between the Software Engineering Institute (SEI) of Carnegie-Mellon University and the U.S. federal government. The goal was to produce a framework for the U.S. federal government to assess the capabilities of its contractors in the area of software development. The first version of the framework, released in 1991, gained rapid acceptance in the defense industry because the Department of Defense used the CMM process maturity level as an exclusion criterion for awarding many of its largest software acquisition contracts.

The introduction of the CMM is a major milestone in the evolution of software development. Most of the tools and methods mentioned previously seek to mitigate the risk of project failure by prescribing analysis, design, and development practices. The CMM instead draws attention to management of the software process. With the CMM, therefore, the relevant question is not what tools or methods are best, but what the most appropriate way of managing the software process is, in order to systematically eliminate defects by improving the process.

2.1.4 The emergence of the consumer software industry

Before the personal computer, software products available on the open market were expensive, typically costing between $5,000 and $250,000. Software packages were developed by professionals and the intervention of salesmen was often required to

sell them because of their perceived complexity. The availability of after-sales support was a criterion considered important in the choice of a software product, and was often offered by the software maker or by authorized distributors. A few hundred sales were generally needed to make a corporate/business software product successful (Campbell-Kelly, 2003: 208-209).

From the early 1980s, with the proliferation of personal computers in homes and schools, a market for personal computer software developed. This had the effect of making software products consumer items (Pugh et al., 1991). Consumer software was considerably less expensive than corporate/business software and sold in much greater volume: it was typically priced between $50 and $500, and several thousands of units were generally sold. Unlike corporate/business software products, consumer software products were more often than not developed by small groups of amateur programmers, and, in some cases, by a lone programmer (Friedman, 1989). The highly-successful database program dBase II, which was written by a moonlighting programmer, is a prominent example.

The personal computer explosion and the emergence of a consumer software industry had a noteworthy effect on the discipline of software development. As explained above, the consumer software industry developed and operated quite independently from the business and scientific software industry. It retained for some time its own rules and logic. Within the consumer software industry, the software crisis was not a tangible reality. As a consequence, programmers did not feel the need to appropriate the development practices of established software producing organizations and remained experimental. In a sense, it can be said that a new way of conceiving and practicing software development spun off the personal computer explosion (Campbell-Kelly, 2003: 227).

Analogies have often been made between the personal computer software industry and the music or book publishing industries. Within the personal computer software industry, software development was principally about releasing a product that appealed to the customer; thus, software development practices had to be customer-focused. In particular, it became good practice to continually improve products incrementally, to release products periodically, and to make products evolve as part

of a product family. Improving products incrementally helped to minimize the risks of releasing products whose features were inconsistent with what customers expected. In a similar vein, releasing products periodically enabled software organizations to remain in constant touch with the market (Gates, 1995: 14-35).

Prototyping became highly topical because it allowed customers to use a software product and give its designers feedback before the software product was released. The importance of prototyping was emphasized in a number of best-selling books and semi-academic publications (e.g., Cusumano, 1998; MacCormack, 2001). Often, Microsoft success stories were used to illustrate the importance of prototyping (e.g. Cusumano and Selby, 1997; MacCormack and Herman, 2000).

Producing families of software represented a radical break from developing 'industrial strength' software. Significantly, the production of families of software appeared better suited to the industrialization of the software development process than the sporadic production of large software. Indeed, producing families of software made code reuse possible on a large scale. Reuse enabled improvement in productivity and quality by incorporating components whose reliability had already been established (Selby, 2005; Wasserman, 1996). OOSD, which happened to promise more effective code reuse, emerged as particularly relevant in the context of the personal computer revolution.

In the mid-1990s, the internet developed into a mainstream medium and began to change the way people interacted (Besser, 1995). A sudden demand for the development of informational and transactional web sites and web-based applications followed, causing rapid growth in the internet sector and related fields. Many entrepreneurially-minded software professionals understood that highly flexible development methods were highly appropriate in the business context created by the internet (Cusumano, 2004; Cusumano and Selby, 1997; MacCormack et al., 2001). Meanwhile, scholars and consultants announced alternative software development models intended to reduce development time and ensure that application met users' expectations (Highsmith, 1997; MacCormack, 2001). One such model, the rapid-prototyping model, supported the development of disposable prototypes intended to establish customer preferences. Significantly, relying on prototyping and the

incremental delivery of functionalities had the effect of reducing the need for structured methodologies. In the internet context, responsiveness and flexibility became the watchwords.

It is important to note, however, that the new software development practices that gained acceptance with the emergence of the personal computer and the internet did not invalidate the more conventional ones. For example, although developing iteratively and making software evolve in the manner of consumer products became recognized as appropriate practices to reduce risk and time to market, methodologies, quality-centric models, and other such applications of Taylor's principles remained widely used. In fact, it seems that it is the development practices of the internet era that lend themselves best to the factory model because they encourage the reuse of components across the different software of a family. To be convinced, one just has to think how game engines, which form the core software component of a computer video game, tend to remain the same for a series of games, or how electronic catalogues are reused across different transactional websites.

While the new software development practices of the personal computer and internet era did not invalidate the more conventional ones, they presupposed different visions of software development. A dissonance seems to have developed at a conceptual level-- that is to say, at the level of how software development should be conceived of. Based on the literature so far reviewed in this chapter, it is quite clear that software development is principally about writing good quality programs on time and on budget, but is it more akin to creative work or factory work? Is software development more akin to a craft or a science? To make things even more complex: can creative work take place in a regimented factory environment? Are craft and science mutually exclusive? The feeling of discomfort in the discipline, vis-à-vis its identity, certainly predates the personal computer and the internet, but it becomes particularly evident when one follows the discipline's evolution-- that is, when changes in practices and beliefs are examined chronologically.

In recent years, confusion about the discipline's identity grew with the expansion of agile methodologies. Agile methodologies are often presented as the solution to the ills that the methodology movement created. In particular, they seek to reduce the

level of bureaucratic activities that the reliance on methodologies tends to entail, encourage a focus on the activities that add value for the customer, and make the release of software more predictable by relying on the adaptive capability of programmers, rather than on rigorous planning. Although agile methodologies often differ considerably in terms of development practices, they all purport to re-establish software development as a creative and people-oriented activity. In 2001, famous members of the software community crystallized the values and principles behind this novel approach to software development by writing and signing the "Agile Manifesto" (see Appendix 3).

Agile methodologies seek to take software development along a totally different trajectory than that it has followed for the past 40 years. Since the first NATO conference, the prevailing belief has been that imposing discipline and rigor upon software development would make this activity more predictable and more efficient. CASE tools, methodologies, and quality centric model have all attempted, in one way or another, to do this. Agilists – proponents of agile development – argue that if development projects often go wrong still today, in spite of the use of sophisticated management and technological innovations, it is chiefly because software professionals have attempted to make an activity characterized by change and emergence predictable. Agilists maintain that software development can become more predictable if risks are mitigated throughout projects by relying on approaches that effectively empower the adaptive capabilities of humans. The key to this, agilists claim, is iterative development.

2.1.5 Open source development

The term 'open source' refers to software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees. Open source software is typically created as a collaborative effort whereby a community of users improves upon the code and shares the changes. From an open source development perspective, the value of software lies primarily in its usefulness to the developer or organization, rather than in the amount of profit it brings to its owner (Peizer, 2006).

The open source development model differs in many respects from the development approaches that have come and gone since the first NATO conference on software engineering: from an open source perspective, software development is neither a formally organized activity, nor an economic activity intended to generate profit. For this reason, some commentators see open source development as a radically new way of conceiving of software development (Raymond, 2000). However, it can be argued that open source development is nothing other than a return to an old approach, to a time when software was developed by its users and exchanged among a community of users. Indeed, from an open source perspective, as in the days when software development was an ad hoc activity, software is valued according to its usefulness or 'use value' (Adler, 2006).

While the vision of software development that the open source development approach entails may be old wine in new bottles, the practices it involves are unquestionably novel. In his 1997 essay, "The Cathedral and the Bazaar," Raymond proposes a model for developing open-source software known as the 'bazaar model.' With the bazaar model, roles remain loosely-defined and development takes place in a decentralized way. The internet is the medium that renders the decentralization of the development activities and the collapse of the divide between developers and users possible. In the same spirit, Robles (2004) argues that the general structure of the software should be modular in order to allow for the parallel development of different releases. In sum, the open source development methodology represents a radical break from more traditional software development approaches, in which people are ascribed roles (e.g., coders, testers, architects) and in which the system design is done by a few architects in order to preserve the architectural integrity of a system.

Advocates of open source development claim that it is superior in a number of ways to the closed source method (DiBona et al., 2000). Raymond (1998) goes as far as to suggest that the open source enables more potential for the development of higher quality software than any other methodology and technique. Critics, however, attribute the popularity of the open source model not so much to the resulting quality of software products, but to the appeal of its anti-commercial rhetoric. The question of whether the open source development methodology is better suited to a particular

context or a particular type of application is still hotly contested, and has recently spurred the development of think tanks and consultancy organizations (Peizer, 2006).

This chapter has, up to this point, mapped the evolution of software development as a discipline by examining the rise and demise of particular ideas and practices. This evolutionary sketch suggests that tools, methods, and methodologies have played a crucial role in defining how software development should be practiced at different points in time. Drawing on the academic literature, the remainder of this chapter will examine how the changing nature of software development is accounted for.

2.2 The formation of beliefs and practices

Hirschheim, Klein and Lyytinen (1996) set themselves the task of delineating the IS community's understanding of information systems development (ISD). The general argument that the authors put forward is that researchers have conceptualized ISD in many different ways because they are relating and interpreting core research results from many schools of thought that differ in terms of research domains and research approaches. These schools of thought, the authors argue, form the intellectual structures of ISD.

For anyone with a basic knowledge of the social sciences, such an argument has a familiar ring to it. Burrell & Morgan (1979) have previously argued that social theory can be conceived in terms of four paradigms, based upon different assumptions with regard to the nature of social science and the nature of society. Before them, Anthony (1965) established fundamental categories according to which the field of decision making and management control was cultivated. Hirschheim et al.'s argument, in spite of being conventional, is interesting because it directly addresses the different ways in which ISD is conceived. It effectively makes us realize that ISD can be conceived in different ways. Moreover, the argument pays due justice to the crucial role that tools, methods, and methodologies play in shaping ISD. Let us take a closer look at the concept they develop.

Although the authors start off the article by focusing on how IS researchers conceive ISD in the academic literature, they quickly bring into their narrative how system

developers conceive and practice ISD. Hirschheim et al. use the terms 'object system class' and 'development strategy' to distinguish between, respectively, the representation of ISD offered by scholars and the social practice of developing IS. The latter term deserves closer attention.

By using the term 'development strategy,' the authors seek to draw attention to the practical character of ISD work. A development strategy embeds a set of consistent and often deep-seated social practices that developers can draw upon while developing an IS:

> [Development strategies] convey enduring and persistent development practices which are gradually solidified from the development experience and learning. Development strategies thereby "crystallize" existing know-how of workable development practices. They are emergent in the sense that they evolve while the programs are instantiated whereby material ISD methods and tools get more sophisticated. (Hirschheim et al., 1996: 27)

The choice of a development strategy, which may be intended or unintended, is fundamentally justified by the desire to achieve some desired outcomes that are in harmony with a set of development 'principles.' Principles, in Hirschheim et al.'s terms, are the "broad normative guidelines and evaluative and behavioral dispositions that underlie the application of specific tools and methods." Development strategies are not developed from scratch, but assembled from tools and methods that provide the rules of actions, resources, and skills required to develop an IS. Principles guide the selection of the concrete methods and tools, and embody the norms "that give legitimacy of using certain tools and methods and thereby certain actions during system development."

The concepts of 'development strategy' and 'principle' suggest that the development activity is based on an actor's underlying set of beliefs and assumptions about the nature of his or her work. Presumably, most system developers see their work as being directed toward the construction of functional IS, but there is a whole spectrum of views on what the task of constructing information systems is. For example, as Lanzara (1983) points out, system development may be seen as solving problems related to the use of information systems based on the assumption that viewpoints

and interests do not change the problem itself, or may be seen as defining and understanding problems related to the use of information systems based on the assumption that the uniqueness of the situation affects the design choice. This sort of nuance, which is inherent in ISD research and practice, leads Hirschheim et al. (1996: 24) to recognize that IS development necessarily occurs by accepting specific development principles that are often ideologically invested-- that is, biased in favour of the use of particular tools and methods.

The first part of this chapter has already presented an overview of the principal tools and methods of the discipline over a period of approximately four decades. A relationship between the manner in which software development is principally thought of and practiced at a given point in time, and the prevalence of certain tools, methods, and methodology has already been observed. Hirschheim et al. offer an elegant explanation for this apparent relationship, but before taking a closer look at this explanation, let us first clarify what the authors mean by tools, methods, and methodologies. 'Methods' are conceived of as prescriptions and rules of action for the development of IS. 'Tools' are the material instruments for the execution of some procedures defined by the methods. When merged together, methods and tools form 'methodologies' (see Table 1). These definitions are not without problems (see, for example, Avgerou and Cornford, 1993), but the researcher accepts them without reservation in order to proceed with more important points.

Tools and methods play an important role within Hirschheim et al.'s thesis in many respects. First of all, tools and methods are integral to the stock of intellectual and material resources that system developers can draw upon while instantiating a development strategy. More specifically, they compose the stocks of knowledge and procedures to carry out the ISD process (see also Baskerville, 1991). Moreover, tools and methods limit the number of ways in which system developers can conceive ISD by drawing attention to only selected aspects of the activity. Hirschheim et al. are, however, careful not to suggest that tools and methods deterministically produce development strategies. Methods and tools can be assembled in multiple ways to produce different development strategies.

Table 1: Theoretical concepts used by Hirschheim et al. (1996)

Theoretical Concept	Definition
Development Strategy	Development strategies convey enduring development practices which are gradually solicited from experience and learning. Development strategies crystallize existing know-how of workable development practices.
Development Principle	Development principles are the broad normative guidelines and evaluative and behavioral dispositions that underlie the application of specific tools and methods that build up the development strategy. Development principles embody the norms that give legitimacy to using certain tools and methods and, therefore, certain practices used during systems development.
Orientation	An orientation represents a consistent set of attitudes, beliefs, assumptions and intentions which a developer brings to the process of IS change. An orientation captures the underlying values, goals and epistemological underpinnings that drive the development activity.
Method	Methods are prescriptions and rules of action for the development of IS.
Tool	Tools are material instruments for the execution of some procedures defined by methods.
Methodology	Methodologies are the combination of tools and methods. Methodologies embody practices and cognitive frames that can be taught, shared and refined in practice.

Following Habermas (1984; 1987), Hirschheim et al. (1996:10) define an orientation as "a consistent set of attitudes, beliefs, assumptions and intentions that developers bring to the process of IS change." In other words, orientations comprise the beliefs and epistemological underpinnings that drive the development activity. So, while the development principles provide rational justification and necessary ideologies for the use of concrete tools and methods, an orientation encompasses the beliefs and assumptions that form an understanding of what ISD is.

Hirschheim et al. submit that there is a two-way relationship between orientations and principles, but the authors do not elaborate on the nature of this relationship. In a few passages, however, the authors imply that it is principally the choice of an

orientation that affects the adoption of principles: "First of all their [the development strategies] content depends on which domain and orientation have been chosen by the actor and how the change is consequently legitimized by accepting specific development principles" (Id.: 24). Elsewhere, Hirschheim et al. also suggest that the choice of orientation is the starting point of everything: "the choice of a development strategy is fundamentally justified by the desire to achieve some of the desired 'outcomes' that are in harmony with the development principles. The framework suggests that the outcomes depend on the actor's dominant orientation" (Id.: 27). Thus, the beliefs, assumptions and intentions – that is, the orientation – that a developer adopts largely determine what ISD is in his or her eyes, as well as what the ensuing development principles should be.

A few examples of implicated issues are, at this point, appropriate to show the myriad of ways in which ISD can be conceived of. Implications for the practice can be identified for each example provided.

- Example 1: Should one see ISD as the construction of an IS or the resolution of an organizational problem? In other words, is ISD a task best left to technical experts? (Hirschheim and Klein, 1989; Mumford, 1983) Mumford & Weir (1979) and Checkland (1981) address issue of participation. They believe that that the intended users of the system should play an active role in defining what the organizational problem to be resolved is.
- Example 2: Should one understand ISD as a rational, systematic activity or an activity in which many of the important details generally become known to system developers as they progress with the implementation? (Boehm, 1988; Markus, 1983; Parnas and Clements, 1986; Stolterman, 1991)
- Example 3: Should one conceive of ISD as a predictable activity, along the lines of factory work? Or should one see ISD as an art or a craft? These alternative conceptions of the activity are discussed in "Japan's software factories: A challenge to U.S. Management" (Cusumano, 1991).
- Example 4: Should one see ISD as an activity that is inescapably bureaucratic? Or should one see ISD as an activity where face-to-face interaction should be favoured in order to keep paperwork to a minimum? (Cockburn and Highsmith, 2001; Highsmith, 2002; McBreen, 2001)
- Example 5: Should one see ISD as an activity geared towards the construction of an IS that meets particular requirements? Or should one see ISD as an activity geared towards the production of business value, even if this involves going beyond the requirements? (Fowler, 2003)
- Example 6: Should one see ISD as a commercial activity that produces fees and profits for an IT organization? Or should one see ISD as an activity that produces business value for the users and their organizations? (Adler, 2006)

- Example 7: Should one think of ISD as a purely-instrumental activity geared toward resolving organizational and/or technical problems? Or should one think of software development as an activity that encompasses an important symbolic dimension; that is, ensuring the legitimacy of the organization in the eyes of both internal and external stakeholders? (Adler, 2006; Avgerou, 2000).

In valuing a particular way of thinking of ISD, the adoption of a certain set of practices is encouraged. For instance, in understanding ISD predominantly as a systematic and predictable activity (see Example 2 and 3 above), people's attention is directed towards the waterfall model and other such plan-driven methodologies. On the other hand, adopting the spiral model makes sense in a context in which it is believed that ISD entails changes to the requirements as projects progress. As the seven examples presented above show, the literature provides ample evidence that beliefs influence how software development should be practiced. If beliefs are so decisive in framing how ISD should be conceived and practiced, how, then, do they become established in an organization? This leads us to formulate the following research question:

Research Question 1: **How do beliefs about software development and software development practices come to be established as legitimate in a software development organization?**

To be sure, as a number of scholars have observed, actors can change their orientations relatively quickly (Dubé, 1998; Dubé and Robey, 1999). Moreover, actors may adopt multiple orientations during ISD, especially if they have to interact with numerous different parties. Consequently, actors involved in a project may very well possess different (and even conflicting) beliefs about the nature of ISD and how it should be practiced.

Divergence in ways of conceiving of software development touches the core of the research. The question that this doctoral study investigates pertains to the perceived validity and appropriateness – that is to say, the legitimacy – of the beliefs about software development. The main objective of this study is to understand the process by which organizational actors go about constituting what developing software means. In particular, the study aims to shed light on how the legitimacy of a vision of software development is negotiated, challenged and, if it is the case, established.

Moreover, this research seeks to refine our understanding of what those beliefs involve in terms of practice adoption.

As noted above, the literature on ISD indicates that beliefs influence how software development should be practiced. This idea is well articulated in Hirschheim et al.'s article. In Hirschheim et al.'s terms, the 'dominant orientation' adopted by developers is largely responsible for determining the 'development principles' underlying the development activity. In other terms, it is submitted that the beliefs that actors bring to the development activity legitimize the use of certain tools and methods, and, therefore, certain practices. The proposition that beliefs influence how software development should be practiced is accepted as true by the research until evidence is found to contradict this proposition.

The management of the process of ISD, the role of methods, and the tasks of software professionals are themes that have remained central in the ISD literature (Mathiassen, 1996). Straddling these three broad themes is the question of balancing the need for creativity and flexibility with the need for operational efficiency and predictability (Boehm and Turner, 2004). The argument is put forth that ISD encompasses many non-routine activities and involves the resolution of emergent issues that require creative thinking and adaptability, but that it can, nonetheless, be made efficient thanks to a certain degree of standardization and formalization (Adler, 2006). Again, how an actor sees ISD along the continuum of creativity/flexibility – efficiency/predictability is very much a question of belief. The following research question is derived from the main research question:

Research Question 2: **How are the contradictory needs for creativity and flexibility, on the one hand, and efficiency and predictability, on the other, negotiated?**

The chronological review undertaken in the previous section sought to highlight the changes that software development has undergone over the past four decades. It should be clear by now that software development was thought of differently in different contexts. The military clearly did not understand software development in the context of Cold War in the same way as business people, prior to the personal computer explosion, did-- or as agilists, in our current internet world, do. In a related

vein, the wide-scale acquisition of tools and methods seems to be associated with some of the distinct ways of thinking about software that prevail in certain contexts. According to Hirschheim et al., the orientation that actors generally adopt in a particular context informs the normative guidelines and evaluative and behavioural dispositions (i.e. the development principles) that underlie the application of specific tools and methods (Hirschheim et al., 1996: 24). It would seem, therefore, that the context has a bearing on the acquisition and local appropriation of tools and methods. Moreover, since these tools and methods are, according Hirschheim et al., instrumental in determining what software development is all about, one can expect the context to influence how software development is predominantly thought of and practiced in an organization. Answering the following research question will help to answer the main research question of this doctoral dissertation:

Research Question 3: **What bearings does the institutional context have on the manner in which software development is principally thought of and practiced in a software organization?**

It is now accepted within organization studies in general, and IS research in particular, that institutional contexts shape organizational activities (Orlikowski and Barley, 2001). Institutionally-informed studies have made us realize that the legitimacy of particular ideas about organization and practices are often influenced by institutional forces. Researchers have provided compelling evidence that the trade literature, popular books, conferences, sales presentations, casual conversations among practitioners, and, most recently, electronic forums provide the channels through which ideas circulate and through which organizational practices are made to evolve (Sahlin-Andersson and Engwall, 2002a). These channels frequently provide a means to introduce and collectively determine how tools and methods for software development can be applied (Swanson and Ramiller, 1997).

An institutional argument runs through this dissertation. It is assumed that in order to understand how beliefs about software development and software development practices come to be established as legitimate in an organization, it is necessary to pay attention to the influence of institutional structures that characterize the organization's context. As will be explained in due course, emphasis is placed on the

normative dimension of institutions (Scott, 2003; Scott, 2001). Attention is directed to the taken-for-granted beliefs that constitute social reality and provide the normative frames that support social practices.

The next chapter presents the theoretical framework used for this study. One of the strengths of the theoretical framework is that it enables the researcher to appreciate the interplay between different levels of analysis. In particular, it enables the researcher to appreciate the manner in which organizational actors go about establishing beliefs about software development and software development practices as legitimate, while paying due attention to the effects of the institutional context in which organizational actors are situated.

3 Theoretical Framework

From the middle of the 20th century, over a period of approximately two decades, sociology took as its principal object of inquiry the function of social structures. During this period, the prevailing belief was that these structures were functional in the sense that they ensured that society would operate smoothly. Emphasis was placed on the norms and value deemed necessary to organize relationships among members of society. Talcott Parsons, the figurehead of sociology during this period, stressed, and placed at the center of analysis, social structures, and ascribed to these structures social functions. Without strong structures, Parsons and his followers argued (Parsons, 1951; 1961), stability and internal cohesion of societies is impossible.

This sociological paradigm, known today as functionalism and structural-functionalism, is a macro sociological paradigm. It gives a great deal of attention to the structures of society. It looks at things on an aggregated scale. It favours the long-term – or a period time sufficiently long to explain the development of patterns of behavior – rather than the episodic. A common criticism directed at functionalism (and other macro sociological paradigms) is that it contains no sense of agency. Within this paradigm individuals are seen more or less as puppets, acting as their role requires. It is criticized for falling short of providing evidence of how intentions orient action without relying on notions such as roles and norms (Elster, 1990). More recently, functionalism has been criticized by conflict theorists, Marxists, feminists and postmodernists for giving far too much weight to integration and consensus, and neglecting independence and conflict (Holmwood, 2005).

The limitation of functionalism led to the development of alternative theoretical approaches to explain how society and organizations can exist in the face of individual interest, and in the 1960's, an ethnomethodology and cognitive revolution was initiated. Harold Garfinkel, a Parsons student influenced as well by the phenomenology of Alfred Schutz, sought to discover the nature of the glue which cements people together in society and the role of cognition in face-to-face interaction. His unique insight was that social order does not derive automatically

from shared norms and social roles, but is constituted, as practical activity, in the course of everyday interaction (Garfinkel, 1967). Also influential on sociology, but coming from outside it, Herbert Simon (Simon, 1957) and his colleagues (Simon and March, 1958) helped accelerate a shift from a functionalist to a cognitive paradigm of organized action.

This revolution marked a shift from a view of the individual as relatively uninteresting entity sleepwalking through interactions to a view of the individual actively engaged in making sense of his or her world. Understanding social phenomena from the perspective of those situated within it became relevant, which required interpretive approaches. Though Parsonian and other strands of functionalist sociology could accommodate the assumption that social structures operated from within the individual, this new sociological paradigm provided the intellectual material needed to articulate the idea that individuals are the creators of their social world. Social structures could be seen as emerging from interactions. And since language is key to mediating interactions among humans, this major intellectual recasting called for language-sensitive approaches (Knorr-Cetina, 1981).

3.1 Organizational discourse theory

This research seeks to bring about a finer understanding of the process by which an organizationally-related object, software development, is constituted. In this sense, the objective of the study is commensurate with the objective of organizational discourse approaches. According to Hardy,

> Scholars interested in the constitutive role of discourse, in one way or another, subscribe to the view that discourse comprises sets of statement that bring social objects into being and, in using the term organizational discourse, refer to structured collections of texts that bring organizationally related objects into being as they are produced, disseminated and consumed. (Hardy, 2004)

Hardy defines an organizational discourse as a "structured collection of texts." A text is generally understood to be a piece of written language. Within organizational discourse studies, a text may be either written or spoken, so that, for example, the words used in a conversation constitute a text. To regard the words spoken as being

text only makes sense considering that those words have to be transcribed in order to be analyzed. Thus, texts can be considered to be a manifestation of discourse and the discursive unit.

Having introduced the notion of organizational discourse, it becomes possible to delineate how it is to be used. The concept of organizational discourse reflects a number of tensions and debates, and it is, therefore, necessary to designate the flavour that is given to it. How the notion of discourse is utilized in this study is largely commensurate with how Ian Parker understands it:

> Discourses do not simply describe the social world, but they […] bring phenomena into sight. A strong form of argument would be that discourses allow us to see things that are not 'really' there, and that once an object has been elaborated into a discourse it is difficult not to refer to it as if it were real. Discourses provide frameworks for debating the value of one way of talking about reality over other ways. […] A good working definition of a discourse should be that it is a system of statements which constructs an object. (Parker, 1992)

An important idea that Parker addresses is that discourses have constituting power. Granting constituting power to discourses provides a basis of the frequently-repeated statement that discourses do not simply mirror the social world, but bring it into being (Rorty, 1967). This perspective on discourse, of course, presupposes a particular conception of the social world. As suggested above, the idea that language has a role in the constitution of reality has become commonplace in a wide segment of the social sciences, primarily as a result of work in social constructionism (Berger and Luckmann, 1967; Wittgenstein, 1953).

In a similar vein, Parker indicates that different ways of talking about something carry different weight. This implies that individuals value discourses differently. An issue that is hotly debated is whether individuals can use discourses intentionally to produce outcomes. The approach adopted in this study rests on the assumption that discourses can be used by individuals in attempts to constitute an organizational object (Grant et al., 1998; Hardy et al., 2000; Phillips and Hardy, 1997). Discourses are presented as resources for debating the value of one way of talking about an organizational object, as opposed to other ways (Parker, 1998).

Seen this way, discourses are constantly valued and evaluated, acclaimed and contested, by individuals who produce and receive them as part of their communicative practices. In other words, they are seen as the object of 'processes of valorization'-- that is, processes by which and through which they are ascribed certain kinds of value (Thompson, 1990). A number of factors affect the value of a particular way of talking about something. For example, the way in which a discourse draws on other discourses, which Fairclough (1995) refers to as interdiscursivity, can influence the impact it will have. Moreover, certain characteristics of actors can contribute to making discourses influential in a given context. One may think, for example, of the resources and capacities of various kinds, such as the rhetorical skills of actors and the formal legal authority granted to actors by an organization.

Although it is believed that discourses can be used by individuals, it is not assumed that individuals have conscious control over the value they collectively ascribe to discourses. As is the case with any other kind of resource, the value of a discourse appears to be largely determined by the context in which it is found. Parker captures this idea by saying that discourses are implicated in the structures of institutions. The concept of institution will be discussed in detail in this chapter, but for the moment, it can be provisionally defined as customs and behavior patterns important to a social group. The term 'institution' is, therefore, taken to mean, 'social institution.' Seen this way, individuals situated in a particular institutional context would tend to value in a similar way a set of statements. It is always the individual who values the discourse, though s/he uses the valuation criteria that the institutional context provides him/her with. Consequently, a discourse that appears perfectly credible in one particular context may be highly questionable in another.

That the value of a discourse varies across contexts suggests that it is not the properties of the text (i.e. content and structure) that primarily determine the value of a discourse. In other words, the value of a discourse does not primarily come from inside its texts but from outside them. For this reason, some theorists contend that the analysis of texts should not be artificially isolated from an analysis of the organizational and institutional context within which texts are embedded (e.g.

Fairclough, 1995; Fairclough and Wodak, 1997). What can be inferred from texts, and how context can and should be used to inform the analysis, is an area of disagreement within discourse theory. The researcher's position is that one cannot properly understand what goes on in any interactional episode unless one knows its place in the relevant institutional context. This point is an important one because it leads the researcher to reject text-centred theories, such as conversation analysis, which represent an important segment of discourse-sensitive studies.

Organizational discourse theory has made important contributions to organizational theory by highlighting how struggles around meanings are played out in organizations. Seen from this broad theoretical perspective, meanings are created and contested as a result of discursive interactions among actors with different interests (Grant and Hardy, 2003; Mumby and Clair, 1997). Consequently, the dominant meaning taken by an organizational object often emerges as alternative discourses are subverted or marginalized. The organizational context appears here characterized by unequal relations of power and populated by individuals possessing different resources.

Parker (1992) believes that institutions are structured around and reproduce power relations, and that "we should talk about discourse and power in the same breath." The point is that the texts in which discourses are actualized are produced, transmitted, and interpreted by individuals who are situated within specific institutional contexts and who possess various kinds of resources. Although the quality and quantity of resources available to individuals can reasonably be presumed to vary from one individual to another, individuals are never outside an institutional context (Fish, 1980). Framed in these terms, this suggests that individuals within a particular institutional context stand in an unequal position of power vis-à-vis one another. That is to say, individuals occupy positions that are marked by different abilities to produce, transmit, and interpret texts, and, hence, different capacities to discursively constitute a social object (Mumby and Clair, 1997). There is, of course,

more to the claim that discourse and power should be handled as joined concepts, but this aspect of the relationship is the one that is relevant within the present study.[2]

This high level discussion has delineated a particular area within discourse theory. This area is primarily concerned with the constitution of an organizational object, rather than with the creation and recreation of the organization. In this respect, the organization is a site of struggle where individuals try to affect and stabilize the meaning of an organizational object. While texts are instrumental in bringing organizational objects into being, they are not presumed to possess agency. Agency is a faculty of human beings and is manifested in one's capacity to produce, transmit, and interpret texts. Thus, certain characteristics of actors within the context of a particular institution will accord them agency. The theoretical framework adopted must, therefore, be power - and context- sensitive.

3.2 Critical discourse theory

Few discourse theorists would admit that their approach is not context-sensitive. The reason most theories can be said to be context-sensitive is that there are many possible interpretations of what counts as 'context,' and how the effects of the context in which a discursive event takes place can be analytically understood. For example, conservation analysts presume that the context is constituted in and through the text produced (e.g. Sacks et al., 1974). Thus, limiting inferences as to what can be validated by reference to what is observable in the texts being analyzed is deemed sufficient to reveal the effects of the context (Heritage, 1984). Admittedly, some discourse analytic approaches are more context-sensitive than others. As such, approaches that explicitly seek to show how discourses and discursive events shape and are shaped by the institutional context are generally regarded as context-sensitive (Hardy, 2004). One of the most influential context-sensitive approaches to the study of organizational discourse has been that of critical discourse analysis (henceforth CDA). Developed by Norman Fairclough (1989; Fairclough, 1995), this approach is distinctive in that it rejects barriers between the study of the micro events and the

[2] Additional aspects of the relationship between discourse and power include the constitution of a subjective identity through discursive practices and the maintenance of a "regime of truth." According to Foucault (1972), a regime of truth is a system of beliefs and values created by a society that enables one to distinguish true and false statements.

macro structures by integrating three layers of analysis. This section focuses on the theoretical propositions inherent to CDA.

CDA is an analytical technique and a set of theoretical propositions for studying language in relation to power and ideology. It is founded on the idea that inferences of an ideological character are pervasive in discourses and discursive practices (see more below on ideology). CDA adopts critical goals: it is designed and presented as "a resource for people who are struggling against domination and oppression in its linguistic forms" (Fairclough, 1995: 1).

As an analytical technique, CDA integrates (a) the analysis of text, (b) the analysis of processes of text production, consumption and distribution, and (c) the socio-cultural analysis of the discursive events as a whole. These three dimensions are respectively referred to as the 'text,' 'discourse practice,' and 'social cultural practice' dimension. CDA is a multi-level analytical technique: by integrating three levels of analysis, it seeks to connect what is going on socially with what is going on linguistically. More specifically, the approach seeks to capture how social structures determine the properties of discourse and how discourses determine social structures (Fairclough, 1995: 27).

CDA is as much a methodology as it is a theory for understanding the operation power and ideology in language use. Power is conceptualized both in terms of asymmetries between actors in discourse events, and in terms of unequal capacity to control how texts are produced, distributed, and consumed in a particular socio-institutional context. An ideology can be broadly defined as a belief that benefits some social class, stratum, or group. A wide range of properties of texts may be considered ideological, including the choice of the words, the formatting devices, and the metaphors invoked. Ideologies may also permeate the norms of interaction that structure discursive events, for example, by determining the turn-taking system.[3]

While the whole of CDA – theory and method – is of value, the researcher is only interested in the theoretical proposition of the analytical framework. Fairclough's

[3] In conversation analysis, "turn-taking" is a process by which interactants allocate the right or obligation to participate in an interactional activity (Sacks et al., 1974).

theoretical propositions offer remarkable insight into how some beliefs and practices come to be regarded as legitimate in a given social setting. CDA (the theory) frames the work of many of the key figures of the social and political thought of the last century, including Louis Althusser, Jürgen Habermas, and Michel Foucault, among others. So, beyond the mere methodology, Fairclough creates a 'state of mind' that can be adopted for clear analysis. Let us now examine these propositions.

Fairclough advances the idea that the language used in speech and writing – what he understands as 'discourse' – may help produce and reproduce unequal power relations among individuals belonging to distinguishable social bases. From this perspective, language is the medium through which asymmetrical power relations are actualized.

Power is here principally understood as the capacity to shape the discourses and discursive practices associated with a particular social domain or institution. It is the stuff that makes an individual or a group of individuals capable of sustaining the legitimacy of particular ways of talking and ways of thinking. As such, the vocabulary chosen, metaphors invoked, and the discourse conventions in force may be indicative of the operation of power.

Fairclough submits that a discourse functions ideologically when it represents some aspects of the social world (i.e. a social object) in a way that sustains or reinforces the privileged position of power of an individual or a group of individuals. Importantly, ideologies are thought to be located as much in the said (the explicit statement) as in the unsaid (the implicit statement). From this perspective, a text that ignores matters that do not benefit those in a privileged position of power may be considered ideologically invested.

Borrowing from Marx, Fairclough goes on to say that ideologies are produced by those who benefit from them. Ideologies are, therefore, not self-generating and do not happen to prevail in a social field or institution simply by chance. However, ideologies may become dissociated to a greater or lesser degree from those who generated them and become adopted by individuals who derive little or no benefit from their prevalence. Such a case arises when ideologies become naturalized.

'Naturalization' occurs when an ideology is largely seen to be commonsensical and based in the nature of things or people. It renders the original reasons for the creation of an ideology invisible. As a process, naturalization involves making what is sectorial, universal and partisan, neutral. In terms of effects, it creates a sort of solidarity between ideology-producers and their antagonists in a social setting. Depending on the degree of naturalization reached, identities, defined in terms of belonging to a group and (ideological) beliefs, may be substantially transformed. Dominated factions may adopt the ideology of a dominating faction as their own without being aware of the forces at play, which may lead to the collapse of differences between individuals and groups of individuals.

Having introduced the concepts of discourse, power, and ideology, as Fairclough understands them, it becomes possible to discuss the critical aspect of CDA. According to Fairclough, the degree of dominance of an ideology is mainly determined by its degree of naturalization; that is to say, by the extent to which the manner in which an ideology serves those in a privileged position of power is masked. From this perspective, a discourse analytical approach capable of making visible implicit propositions of an ideological character that are present in discourse has the potential to denaturalize them.

Adopting a critical goal, Fairclough asserts, "means aiming to elucidate such naturalizations, and more generally make clear social determinations and effects of discourse which are characteristically opaque to participant" (Fairclough, 1995: 28). In the same vein, the author adds that "[t]o denaturalize them [ideologies] is the objective of a discourse analysis which adopts 'critical' goals." (Fairclough, 1995: 27). Domination and oppression in its linguistic form should disappear as ideologies are denaturalized.

This doctoral research does not follow a critical agenda-- at least, not in the sense of aiming at liberating individuals from domination and oppression. If care is taken to explain the critical 'philosophy' on which CDA is based, it is because it leads to two crucial theoretical propositions: that the micro and macro level of analysis have to be integrated, and that institutions have to be the pivotal point between those two levels. Let us consider the following statement:

> The critical approach has its theoretical underpinnings in the views of the relationship between 'micro' event […] and 'macro' structures which see the latter as both the conditions for and the products of the former, and which therefore reject rigid barriers between the study of the 'micro' (of which the study of discourse is part) and the study of the 'macro.' (Fairclough, 1995: 28)

Why does a theoretical approach that is critical have to account for the relationship between the micro and the macro? The message Fairclough is trying to convey is that a theoretical approach that is 'critical' has to go beyond describing that which is immediately visible in texts and interactional episodes, and address the deeper causes of things. Fairclough sees descriptive approaches-- approaches that focus exclusively on the content and structure of the text-- as inadequate for explaining why an interactional episode unfolded as it did and what its effects are. Given the effects of ideologies, the analyst must step in and bring his or her understanding of the wider context into the analysis:

> … the concept of ideology is incompatible with the limited explanatory goals of the descriptive approach, for it necessarily requires reference outside the immediate situation to the social institution and the social formation in that ideologies are by definition representations generated by social forces at the levels. (Fairclough, 1995: 45)

But CDA also goes beyond the micro (and that which is descriptive and 'uncritical') by seeking to understand 'how things are,' and by extrapolating how micro interactional episodes cumulatively constitute the macro institutional context.[4] The author notes:

> … descriptive work generally has been little concerned with the effects of discourses. And it has certainly not concerned itself with effects which go beyond the immediate situation. For critical discourse analysis, on the other hand, the question of how discourse cumulatively contributes to the

[4] So, although discourse analysis approaches are generally associated with a social constructivist ontology, critical analytical approaches (including CDA) are generally rooted in a realist ontology in that they acknowledge the existence of a reality existing outside human consciousness. In a recent article, Fairclough (2005) stated his position as follows: "From the perspective of the realist view of discourse I have outlined [i.e. CDA], it makes little sense to see […] agency and structure … as alternatives one has to choose between."

> reproduction of macro structures is at the heart of the explanatory endeavour. (Fairclough, 1995: 43)

Hence, in Fairclough's view, a theory that is critical is a theory that makes visible the interconnectedness of things. CDA is considered 'critical' because it has the power to reveal how the macro influences the micro and how the macro is reproduced through micro events. Furthermore, an approach that is critical does not stop at answering the 'what' and 'how' questions, but has to look into the causes of things by answering a 'why' question.

The question of how the macro influences the micro is highly relevant to this doctoral thesis. The researcher realized at a very early stage of the research that a theoretical framework that takes into account the influence of institutional forces was required to better understand how beliefs about software development come to be socially established as legitimate. So, for the moment, the term 'macro' is broadly understood as 'institutional.' This treatment of the macro will become more precise as the doctoral thesis progresses. However, it is important to note that how the micro creates and recreates the macro is outside the scope of this doctoral thesis. The research clearly does not seek to develop an understanding of how micro-episodes of social action (e.g. discursive events) in an organization contribute to creating and recreating a largely- shared, perhaps even institutionalized, way of thinking of and practicing software development outside the study organization.

Having examined the rationale for using a theoretical framework capable of taking into account the relationship and tension between the micro and the macro, let us look at the claim for considering the institution as the median point between the two levels. Referring to questions concerning the naturalization of ideologies, Fairclough writes:

> My reasoning is in essence simply that (a) such questions can only be broached within a framework which integrates 'micro' and 'macro' research, and (b) we are most likely to be able to arrive at such research integration if we focus upon the institution as a 'pivot' between the highest level of social structuring […] and the most concrete level, that of a particular social event or action. (Fairclough, 1995: 37)

In this quotation, it is not clear what "highest level of social structuring" can possibly be. The reader is left to figure out what operates above the institutions. In any cases, CDA falls short of accommodating a level which is much higher than that of the institution. Indeed, the 'social cultural dimension' of the model appears quite similar to what is normally understood as the 'institutional dimension' within organization studies (Scott, 2001; Zucker, 1987).

A close look at the theoretical approach reveals that Fairclough actually assumes that the socio-cultural context is composed of a number of institutions that crisscross each other. Moreover, the distinction between socio-cultural context and institutions does not appear so important. The author often talks of the socio-cultural context and institutions in the same breath without making a clear demarcation between them. What is original, however, is how the author defines an institution (or a social institution).

In keeping with the linguistic approach to the study of power and ideology he advocates, Fairclough defines a (social) institution as an "apparatus of verbal interaction." From this perspective, a social institution is an entity that possesses its own conventions or norms of language use. A social institution also defines the social identities and relationships of its members, the topics of importance, and the goals to be achieved.

Fairclough insists that a social institution simultaneously constrains and enables the verbal interactions of its members. A social institution provides its members "with a frame for action, without which they could not act, but it thereby constrains them to act within that frame" (Fairclough, 1995: 38). Moreover, it should be noted that such an institutional frame consists of formulations and symbolizations that are ideologically invested, with formulation referring to a particular way of talking, and symbolization to a particular way of representing and seeing a social object. So, in keeping with the definition given above, *institutions are apparatus of verbal interaction that sustain a set of ideological and discursive norms*.[5]

[5] This definition is by no means final. It will later be explained that an institution houses many different sets of ideological and discursive norms.

Fairclough submits that an institution typically provides alternative sets of discursive and ideological norms. In other words, there is, according to the author, a one-to-many relationship between the institution and the ideological positions that its members may adopt. Thus, a given institution may accommodate two or more distinct ways of talking about and seeing a social object. This diversity of ideological positions, Fairclough notes, "is a consequence of, and a condition for, struggles between different forces within the institution." (Fairclough, 1995: 40). In keeping with the neo-Marxist tenet, this struggle is presumed to be connected to class struggle. And at stake in the struggle between classes is the ideological and discursive control of the institutions itself.

The idea that an institution accommodates a plurality of discursive and ideological norms becomes particularly well articulated as Fairclough deploys the concept of "ideological-discursive formation" (IDF). An IDF, as the term indicates, is a sort of coherent ensemble of discourses that sustain a particular way of talking about a social object that is of interest within a social institution. An IDF favours a particular vocabulary, or more generally, a language for talking about something – for example, the language of science, of Christianity, of management.

The concept of IDF serves two purposes. First and foremost, it cements the relationship between a way of seeing and a way of talking. The logic is that by structuring what can and should be said within a particular social-institutional context, an IDF structures how a social object can be represented – hence, the complex ideological-discursive. So, strictly speaking, it is not the institution as such that sustains an ideological representation, but an IDF contained within the institution. This leads us to the second role the concept of IDF is made to play within Fairclough's intellectual arsenal: the role to depict institutions as being ideologically fragmented.

It is the concept of IDF that makes convincing the claim that a given institution may house two or more distinct ideological positions. *Implicit in this claim is the theoretical proposition that institutions are pluralistic rather than monistic*. Furthermore, it is Fairclough's contention that it is the plurality of IDFs that creates class division within an institution and fuels the struggles between classes. That said,

one cannot help wondering why Fairclough chooses not to see classes – classes, it must be emphasized, that possess their own discursive and ideological norms – as institutions in their own right. A probable answer is that these classes are intertwined in linguistic interactions – a typical case of 'bourgeois meet proletarians.' After all, if classes were not in relationship to one another, domination and oppression could not occur, nor could struggles ensue.

Fairclough's depiction of institutions as fragmented entities stands in marked contrast to the conventional treatment of institutions within the social sciences in general, and organization studies in particular. When the concept of institution is invoked in organization studies, it is more often than not to accent social order and consensus, rather than contradiction and conflict (Scott, 2001). The proposition that a social institution houses many IDFs is interesting, for the full explanatory power of the theoretical approach becomes discernible.

Thus empowered, Fairclough goes on to say that IDFs within a social institution are ordered in dominance. Discursive practices, he notes, "are characteristically ordered in dominance in the sense that there may be a dominant ('normal,' naturalized) practice and dominated (marginalized, 'alternative') practice (Fairclough, 1995: 12). In this sense, the struggle that occurs within the institution "can be seen as centering upon maintaining a dominant IDF in dominance (from the perspective of those in power) or undermining a dominant IDF in order to replace it" (Fairclough, 1995: 41). It is when an IDF has undisputed dominance in an institution that the ideological and discourse norms of the IDF become the most naturalized. In such cases, the norms of the IDFs may come to be seen as the norms of the institution itself.

The limitations of Fairclough's theoretical propositions will be discussed in detail at the end of this chapter. One of the main problems that the author of this dissertation sees with CDA is that it does not possess the analytical constructs needed to capture and contextualize relevant factors that fall outside the immediate remit of the linguistic. The theory of symbolic power and its afferent concepts are introduced at this point as an addition to the theoretical framework being devised. It is the researcher's contention that the notion of symbolic power and the dual concepts of

'capital' and 'field' will empower substantially the analysis by creating an indissolvable bond between the discourses and the institutional context.

3.3 Symbolic power

Born in 1930 in Béarn, a province in Southern France, Pierre Bourdieu received a degree in philosophy before moving into anthropological and sociological research. While serving in the French Army in Algeria, Bourdieu carried out ethnographic work that led to the publication, in 1958, of a book about the Kabyle society. The research in Algeria formed the basis of much of his subsequent theoretical writing, most notably, "Outline of a Theory of Practice" (Bourdieu, 1977) and "The Logic of Practice" (Bourdieu, 1990).

Bourdieu's theory of practice is founded on the idea that the deep-seated practices of individuals (including their linguistic practices) reflect the social condition within which these practices were acquired. Individuals following a comparable life trajectory should face similar challenges and respond to them with similar actions. Bourdieu is particularly interested in how homogeneity among individuals of similar backgrounds creates an observable heterogeneity across groups of individuals of different backgrounds, and how these differences are exploited knowingly and unknowingly by individuals. The author describes the pervasive production, reproduction, and exploitation of the systems of difference as 'symbolic power' (or, in some cases, as 'symbolic violence'). This section introduces the notion of symbolic power and afferent terms.

The notion of symbolic power is a rather flexible one which is used in many different ways throughout Bourdieu's writings. However, one interpretation that stands out is the power of constituting the taken-for-granted through utterances, "of making people see and believe, of confirming or transforming the vision of the world and, thereby, action on the world and thus the world itself" (Bourdieu, 1991: 170). It enables one to obtain without physical or economic force, thanks to the acceptance and collaboration of (dominated) individuals, a standard vision of the world that is suited to those occupying privileged positions.

As the foregoing quotation illustrates, Bourdieu uses the word 'world' abundantly when speaking of symbolic power, but, in fact, what he means is "the objects of the social world" (e.g. Bourdieu, 1989: 20). This precision may appear trivial, but it is important to making clear the fact that the theory of symbolic power may be applied to understanding any kind of social object – not just the world as a whole. Such a social object may be a specific thing, individual, state of affairs, and so on. It is also worth noting that, as the previous quotation also suggests, symbolic power not only affects perception, but also affects action. Depending on how an object is perceived, the ensuing action on the objects of the social world may differ.

The notion of symbolic power implies four key characteristics. First, it is a consensual notion of power in that it involves the willing participation of individuals to accept of particular vision of the world: "Symbolic power is that invisible power which can be exercised only with the complicity of those who do not want to know that they are subject to it or even that they themselves exercise it" (Bourdieu, 1991: 164). It never involves economic or physical coercion. Secondly, symbolic power is a relational notion of power. Symbolic power does not reside in a particular symbolic system or do anything by itself. Rather, it is constantly "defined in and through a given relation between those who exercise power and those who submit to it…" (Bourdieu, 1991: 170). Thirdly, symbolic power is pervasive. Symbolic power is not a specific type of power, but rather an aspect of most forms of power as they are routinely deployed in social life (Bourdieu, 1989: 23). Finally, symbolic power is insidious, as its degree of efficacy is directly proportional to its ability to disguise itself. When symbolic power is exposed, it evaporates. This last characteristic deserves a deeper consideration, as it touches on the concept of 'misrecognition'.

In Bourdieu's view, when power that could be excised through physical force becomes transmuted into a symbolic form, it acquires a legitimacy that it would not otherwise have. The author expresses this point by saying that symbolic power is 'misrecognized,' as such, and thereby 'recognized' as legitimate. The term 'recognition' underscores the idea that the exercise of power through symbolic means always rests on a foundation of shared beliefs (or knowledge)-- that is, on an understanding and acceptance of an institutionally-organized symbolic system. This fact means that symbolic power resides in, and is sustained by, the very structure of

the institution in which the beliefs are produced and reproduced. In actual fact, it is by adjusting itself to the structures of an institution that symbolic power becomes misrecognized.

The French scholar wrote extensively on how the mental structures of social agents (i.e. "subjective structures") become aligned with the structures of the institutions (i.e. "objective structures"). According to Bourdieu, social agents situated within a particular institutional context tend to perceive the institutional constraints as neutral because their minds (their cognitive structures) are issued out of the structure of the institution. This leads social agents to misrecognize their condition as being natural and based on the natural order of things. Misrecognition thus subtly guarantees the cooperation of the dominated in the maintenance of the very institutional structures that may disadvantage them.

For example, using the case of the French education systems, the author illustrates how teenagers raised in a working class environment, who tend to be economically and culturally disadvantaged in comparison to those raised in professional class environment, are led to leave educational institutions relatively early to take on manual work. In this case, symbolic violence occurs when working class teenagers come to assume that their limited academic prospect is due to a lack of scholastic ability, rather than to the maintenance of a system that makes access to and success within academia more difficult.

Like Fairclough, Bourdieu uses the concept of 'class' to refer to a group of social agents who possess similar attributes and derive common benefits from the production and maintenance of a particular representation of the social world or a social object. For both scholars, classes remain theoretical constructs produced by analysts to explain or depict observable social phenomena. The use of the concept is, however, more central and complex in Bourdieu's theory of symbolic power, as class positions are determined by the volume of 'capital' possessed by social agents and the relative value granted by the 'field' to the different forms of capital. The concepts of 'capital' and 'field' are presented in the sections that follow.

3.3.1 Capital

Capital is at once the resource available to social agents to impose or inculcate a vision of a social object, and a marker of social status. There are three principal forms of capital in Bourdieu's writing. 'Economic capital' includes money and other material assets. 'Cultural capital' includes academic credential, knowledge, and skills. 'Symbolic capital' refers to the other various forms of capitals perceived and recognized as legitimate.

Because of the discourse-sensitive theoretical approach adopted herein, it is necessary to at least introduce the notion of 'linguistic capital.' Bourdieu posits that social objects are constituted through a wide array of practices. The notion of linguistic capital is designed to draw attention to the role of discursive practices in constituting social objects. Linguistic capital is understood as the competence, or resource, required to constitute a social object compellingly, and to change or reaffirm the practices of individuals in relation to the social object (Bourdieu, 1991: 37 & 56). In order words, linguistic capital is the resource that enables social agents to make the texts they produce and diffuse consequential.

Bourdieu is careful not to let his readers assume that the quantity of linguistic capital a person possesses is determined by the quality of the person alone (Bourdieu, 1991: 73). As such, the competence to constitute a social object through discursive practices must not be equated with rhetorical skills or the technical mastery of a language in a strict linguistic sense (i.e. the knowledge of a body of words and the rules for combining them). According to the author, linguistic competence is determined by a number of institutional factors, including the institutional role of the text producer (e.g. his title) and his practical understanding of the norms regulating discursive practices that reside outside the purely linguistic aspect of language: "The competence adequate to produce sentences that are likely to be understood may be quite inadequate to produce sentences that are likely to be listened to […]" (Bourdieu, 1991: 55).

For Bourdieu, authority, or the ability to constitute the given through words, does not reside in the words that one utters, but in the power that is endowed to an individual by an institution or field. The author expresses this point as follows:

> […] authority comes to language from the outside […] The efficacy of speech does not lie in 'illocutionary expressions' or in discourse itself […] for it is noting other than the delegate power of the institution. (Bourdieu, 1992: 147)

Like the three other forms of capital introduced above, linguistic capital contributes to establishing the status of a social agent in a given social setting. All other things being equal, the more linguistic capital an individual possesses, the more s/he is able to represent a social object in a way that suits his or her interests and, hence, exert symbolic power. Seen this way, linguistic exchanges are not merely relations of communication between a sender and a receiver, but also relations of symbolic power. According to Bourdieu, "linguistic relations are always relations of symbolic power through which relations of force between the speakers and their respective groups are actualized in a transfigured form. […] Even the simplest linguistic exchange brings into play a complex and ramifying web of historical power relations between the speaker, endowed with a specific social authority, and an audience, which recognizes this authority to varying degrees" (Bourdieu, 1992: 142-143). But the value of a form of capital is not absolute. The value granted to the different forms of capital varies across institutional settings, or 'fields.' To be more accurate, it is the field that determines the value of a certain form of capital. We see here how the notions of capital and field are interconnected.

3.3.2 Field

Bourdieu has been praised for his effort to move beyond a series of oppositions and antinomies that are well embedded within the social sciences (e.g. Ritzer, 1996). For anyone involved in the social sciences today, these oppositions have a familiar ring: agency versus structure, micro versus macro, subjectivism versus objectivism, etc. The dual notions of 'field of forces' and 'field of struggle' are one instance of the care the author takes to dissolve such oppositions by integrating different levels of understanding.

In the course of an interview with Loïc Wacquant, Bourdieu provided us with a concise explanation of the notion of 'field' and its place in his thinking. A field,

according to Bourdieu, is a social space where manoeuvres take place over specific stakes. To think in terms of field involves recognizing the centrality of social relations in social analysis:

> I define a field as a network, or a configuration, of objective relations between positions objectively defined, in their existence and in the determination they impose upon their occupants, agents or institutions, by their present or potential situation […] in the structure of the distribution of power (or capital) whose possession command access to the specific profits that are at stake in the field […] (Bourdieu and Wacquant, 1989: 39)

When viewed as a 'field of forces,' the field is a space with its own logic, rules and regularities. The field of forces is shaped by the structure of the existing balance of forces between different forms of capital-- that is to say, by the relative value it attaches to the different forms of capital. When viewed from this angle, the field structures, yet without determining. The field of force presupposes, and generates by its very functioning, the belief in the value of the stakes it offers. The field of force aligns the action of individuals who enter it in the pursuit of its cause through the means it imposes on them. John B. Thomson, one of Bourdieu's principal commenters in the English-speaking world, explains:

> The very existence and persistence of the […] field presupposes a total and unconditional 'investment', a practical and unquestioning belief, in the game and its stakes. Hence the conduct of struggle within a field […] always presupposes a fundamental accord or complicity on the part of those who participate in the struggle. (Thompson, 1991: 14)

When viewed as a 'field of struggle,' the field is a social space characterized by conflict. It is a site on which social agents' strategies are concerned with the preservation or improvement of their positions with respect to the defining capital of the field. Within economic organizations, this struggle may take the form of a clash between occupational groups, for instance, "between production and publicity, between engineering and marketing" (Bourdieu, 1984: 309). The field is dynamic in that the outcome of a struggle may change the structure of a field – ultimately determining which forms of capital are subordinated to others, and which groups of capital holders (i.e. class) are dominant.

Having presented the notion of field, it is appropriate to make two observations on the notion of field. First, like the notion of class, the notion of field is an analytical construction. While there are cases in which the boundaries of the field are legally determined, such boundaries are often imprecise and generally require an empirical investigation to be determined (Bourdieu and Wacquant, 1989: 39). Within the context of this doctoral research, the notion of field is used to refer to the unit of an economic organization. As will be justified later in this chapter, this unit can be considered a field in its own right because it has its own particular stakes and rules.

Moreover, the notion of field allows for accounting of agency and remains open to the possibility of change. In a field, social agents constantly work, collectively or individually, to differentiate themselves from their closest rivals in order to reduce competition. They sometimes succeed, which enables them to impose and inculcate an alternative vision of a social object as legitimate. Hence, the theory of symbolic power is not merely about reproduction and stability. This is where the theory of symbolic violence differs most from conventional structuralist theories that pay little attention to the role played by social agents and the possibility of discontinuity.

3.4 Critical synthesis

In this doctoral dissertation, organizational discourse theory is taken as the primary theoretical lens. Organizational discourse theory, as explained above, is concerned with understanding how organizationally-related objects are brought into being, as texts embodied in the practices of talking and writing are produced, disseminated, and consumed. From this perspective, discourse is the principal means by which organizational members create their social world, and language is the instrument that enables them to do so.

Within the majority of discourse analytical theories, discourse is seen as being central to the social construction of reality (Phillips and Hardy, 1997). Indeed, a frequently-repeated statement is that discourses not only merely represent the objects of the social world, but constitute them. Such an assumption, at first sight, appears to betray a blatant allegiance to a strongly constructivist ontology: objects of the social

world are not known for what they are, but for what they are perceived to be. Seen in the light of this ontological approach, only observational claims matter. From there, an array of all-too-familiar criticisms can be generated, including, of course, that organizational discourse theory denies the existence of a 'real' world and the effects of social structures (Guba and Lincoln, 1994).

However, if one dares to dig deeper, it quickly becomes obvious that different approaches within organizational discourse theory focus on different aspects of the process whereby the social world is brought into being. Phillips & Hardy (2002) call attention to the differences in the degrees to which researchers focus directly on the dynamics of power, and differentiate between 'constructivist' and 'critical' discourse studies. Critical studies are distinctive in that they recognize that discourse embodies structures of power and ideologies (Hardy, 2004). Whereas constructivist studies are primarily concerned with explaining 'how' social interactions mediated through language produce objects, critical studies seek to uncover 'why' an object is constituted predominantly in one particular way, rather than in another.

With CDA, an object does not emerge out of interactions occurring in some kind of contextual vacuum. Understanding and examining power dynamics and the ideologies that characterize context are important to tackling the 'why' of a social phenomenon. Fairclough (2005) clearly spells out his ontological position in these terms: "My position is an ontological realist one: the social world is indeed a socially (and in part discoursally) constructed world, but at any point in time people are confronted with a pre-structured world which has real properties and a real structure which cannot be reduced to, and are unconditionally subject to, people's knowledge of it …"

In this doctoral dissertation, it is recognized that discourses embody ideologies and structures of power. The constitution of an organizational object is not presumed to be neutral. However, unlike most critical discourse studies, this dissertation does not aim at liberating organizational actors from the unnecessary constraints that prevent their development as individuals – a concern at the heart of the 'critical' project (Alvesson and Willmott, 1992). CDA is here stripped of its radical edge by seeing

'ideologies' as 'beliefs.' Consequently, any group has its particular beliefs corresponding to its position in social life.

In a same vein, the notion of power which carries a pejorative connotation in the writings of Fairclough – for it is the cause of domination and oppression – is toned down and taken the way Bourdieu has taken it. Although Bourdieu is concerned with issues of domination and oppression, symbolic power remains principally an analytical concept. The theory of symbolic power is not designed to change the world in practice or in theory. As one of the French sociologist's long-term collaborators observes, Bourdieu has "always studiously kept aloof from anything that marches under the self-proclaimed banner of 'radical' sociology or 'critical' theory" (Bourdieu and Wacquant, 1989: 192).

Bourdieu believes that science is still the best tool available for the critique of domination, but he also believes that scientists tend to overestimate their capacity to elide domination in practice. Bourdieu, in fact, reckons that the involvement of scientists in struggles against domination often mask their specific interests-- that is, the interests of the social class to which they belong (Bourdieu, 1992: 193). Consequently, using science as a weapon to fight domination is considered contributing to the reproduction of relations of power, rather than to the elision of power. The author of this dissertation adopts Bourdieu's critical attitude towards theories explicitly designed as a resource for people engaged in social action.

In terms of limitations, CDA focuses on the use of language to an extent such that it is inclined to reduce social life to a way of staging texts. Although Fairclough contends that CDA is as much about the use of language as it is about the study of power and ideology, the fact remains that the theory is based and depends entirely on an analysis of language. As Fairclough overtly acknowledges, CDA is nothing other than "an analytical framework – theory and method – for studying language" (Fairclough, 1995: 1). This myopic view of texts and the way they are produced, disseminated, and consumed may easily lead analysts to miss important details that can be more easily captured and understood by other means-- for example, through an ethnographic analysis of social structures and settings. Consequently, Fairclough constantly has to defend the claim that "language is not just a way of staging a text

[but also …] involves particularities of 'field' – what social practices are referred to and how they are signified, of 'voice' – who the participants are […], of 'style' – how participant relations are constructed, and of 'mode' – what forms of textualization and of text-context relation apply" (Fairclough, 1995: 14).

However, CDA falls short of providing the analytical constructs needed to attend to such particularities. Although the problems associated with concentrating on the use of language can be easily resolved at the level of method by bringing together CDA and an in-depth immersion in the field, the lack of analytical constructs needed to understand and explain how things are, and why they are as they are, remains problematic. To put the matter differently, CDA does not possess the analytical constructs needed to capture and contextualize relevant factors that cannot be captured in terms of discourses and discursive practices. That is why making use of the theory of symbolic power and its afferent concepts is appropriate.

The theory of symbolic power does not focus on the use of language to the extent discourse theoretical approaches do, yet it remains discourse-sensitive. The thrust of the argument Bourdieu lays out is that language is the vehicle of power relations, rather than a mere means of communication, and that it must be studied within the interactional and structural contexts of its production and circulation. Moreover, Bourdieu offers the analytical constructs to do so.

To be fair, one cannot criticize Fairclough for not delivering what he promises. CDA is not intended to be a set of logically-interconnected propositions framed in terms of precise, unambiguous concepts. The theory is particularly ill-suited to a logical and propositional reading of the world. What CDA is meant to do is to communicate a certain posture, to offer a certain way of looking at the world in order to change it – and it does that very well. CDA and the theory of symbolic power are highly commensurate. Although with some nuances, both theories seek to connect the micro and the macro in order to gain an understanding of why things are the way they are; both theories take seriously the roles and effects of institutions; and both theories recognize that some individuals are more able than others when it comes to determining how the objects of the social world should be seen and represented. However, the analytical constructs are generally more thoroughly-articulated in the

writings of the French scholars, and come to form a whole. Table 2 presents the main analytical constructs that compose the theoretical framework. These constructs inform the research by providing the vocabulary and proposition necessary to understanding and representing the social phenomenon observed.

Table 2: Key theoretical constructs

Theoretical construct	Definition
Organizational discourse	Structured collection of spoken and written texts that bring organizationally-related objects into being as they are produced, disseminated, and consumed.
Ideological-discursive formation	A complex of (organizational) discourses that exist within an institution and that correspond to a particular "way of talking" and "way of seeing."
Institution	An apparatus of verbal interaction that sustains a set of ideological and discursive norms.
Symbolic power	The power to confirm or transform the manner in which agents see a social object and, thereby, the power to change and reaffirm the practices of agents in relation to a social object without physical or economic force.
Capital	A marker of social status. The resource drawn upon to change or reaffirm beliefs and representations.
Field of force	A social space with its own logic [stakes] and rules whose structure is determined by the value it attaches to different forms of capital.
Field of struggle	A social space where agents seek to improve their relative position of power by legitimizing particular ways of seeing and engaging with a social object.
Apparatus	A field where the dominant has annulled the resistance and reaction of the dominated. A field where the effects of domination are such that changing the structure of the field is impossible.

So far, it is the limitations of CDA that have been scrutinized. The theory of symbolic violence, it is argued, can help to alleviate and resolve some of those limitations. However, the theory of symbolic violence has itself a number of limitations that cluster around two broad themes. These are the emphasis the theory

places on the effect of structures and on social reproduction at the cost of neglecting agency and social change.

The limitations of the theory of symbolic violence are, in part, attributable to its structuralist roots. Structuralism is a theoretical perspective and methodological approach in contemporary social sciences concerned with understanding how language builds upon some higher mental, linguistic, social, or cultural structures (Manning and Cullum-Swan 1994: 467). In terms of methodology, experience is secondary to the deep structures by which meaning is produced and reproduced within a culture.

Structuralism had its moment of glory on the French intellectual scene and helped bring about the "linguistic turn" in social theory. Structuralism, however, became increasingly criticized from the early 1960's for being ahistorical and for favouring deterministic structural forces over the ability of individual people to act. The student uprisings of May, 1968 led French scholars to pay greater attention to issues of power and political struggle, and helped accelerate the transition from structuralism to post-structuralism.

3.4.1 Agency

Post-structuralism has been criticized for allowing little scope for agency, for reducing individuals to passive entities subject to the effects of social structures and institutions. According to Jenkins (2001), the theory of symbolic violence has little to say about the motives of individuals as agents. Moreover, the extent to which individuals are capable of making choices and imposing these choices on the world appears, within the confines of the theory, relatively limited. However, the theory is not completely devoid of any sense of agency.

It cannot be denied that Bourdieu is only moderately interested in the way agents think about, account for, or represent the objects of the social world. The theory of symbolic violence was developed as part of a concern for better understanding what make the practices of social agents, and invites analysts to examine the structures capable of guiding or constraining the agents' practices. However, the theory of

symbolic violence maintains that individuals are actively involved in a struggle over the symbolic representation of the world. Thus, agency and the agent clearly matter. Moreover, resistance plays a central role in the theory of symbolic violence. Defending himself from the accusation of leaving little room for agency, Bourdieu makes the following point: "I cannot begin to comprehend how relation of domination, whether material or symbolic, could possibly operate without implying, activating resistance" (Bourdieu and Wacquant, 1989: 80).

In sum, the theory does not celebrate agency to an extent that would satisfy scholars accustomed to working in the tradition of Schutz's phenomenology, Blumer's symbolic interactionism, or Garfinkel's ethnomethodology. The theory does not take as its starting point how individuals perceive and construct the objects of the social world, but rather what the practices (including discursive practices) of individuals are, and how these practices are animated and constrained by structures. Bourdieu labels his own orientation 'constructivist structuralism' (or "structural constructivism").

3.4.2 Change

Bourdieu's structural constructivism is an orientation which emphasizes the regulatory character of social life. It attributes to individuals a propensity to reproduce the social structures and accept their existing condition. The theory of symbolic violence offers an explanation of how the structures of power are reproduced, as well as why societies tend to hold together, rather than fall apart.

Numerous scholars have criticized the theory of symbolic violence, and other of Bourdieu's models, for leaving little room for the change and the irruption of history (Collins, 1981; DiMaggio, 1979; Jenkins, 1982). Giroux (1983: 92) asserts that, for Bourdieu, "working-class domination … appears as part of a Orwellian nightmare that is irreversible as it is unjust." Bourdieu's skeptical position regarding the capacity to bring about change may, indeed, appear fatalistic.

Once again, Bourdieu falls back on the notion of resistance to show that the theory of symbolic violence is open to change, and is not ahistorical. For him, resistance is

fuelled by a desire for, and a capacity to, change. However, Bourdieu does not ignore the fact that resistance and the status quo often coexist within deterministic models of social reproduction, as was made obvious by Marx. For Bourdieu, if change were to happen, it would be radical, as foreseen by Marx. In this sense, the theory of symbolic violence does not say much about ongoing adaptation and evolution.

In sum, the theory of symbolic violence and CDA have their own strengths and weaknesses. CDA presents the potential to understand how an organizational object, such as software development, is constituted and why it is constituted the way it is. In essence, it draws attention to the relationship between the micro and the macro, and the ideologies at work within the institution in which social agents are situated. However, this theory may tend to focus too much on the textual character of social life and be overconfident of its power to change the social world for the better. The theory of symbolic violence, on the other hand, offers powerful analytical constructs to understand the relationship between the micro and the macro, but only superficially addresses the notion of change.

The theory of symbolic violence and CDA share many similarities. They are both concerned with understanding power relations. They both share an interest in illuminating how beliefs come to be accepted as legitimate and be reflected in the practices of actors. More importantly, they both recognize that studying the use of language can provide insights into the process by which an object, such as software development, is constituted.

As argued above, organizational discourse theory is highly appropriate to investigating how beliefs are negotiated in organizations and how these struggles shape organizational practices (Grant and Hardy, 2003: 6-7). Taking organizational discourse theory as an anchoring point, the theory of symbolic violence and CDA are combined to form a theoretical framework to study the constitution of software development. It is believed that conducting an analysis in terms of 'field of struggles,' 'field of forces,' and 'capital' can enable a theoretically-rigorous analysis of the process by which beliefs about software development and software development practices come to be established as legitimate in an organization. Such

a Bourdieuan analysis will benefit from several of Fairclough's ideas, such as the idea that an institution functions as an "apparatus of verbal interaction".

The nature of reality and the purpose of research have been in the background of the discussion throughout this chapter. The next chapter will delve into the philosophical beliefs adopted for this study and explain how the empirical study was conducted in order to answer the research questions.

4 Research Methodology

This chapter presents the philosophical assumptions that underlie the present study and the methods employed to collect and interpret data. Primary emphasis was placed on selecting methods effective at capturing and preserving the depth and richness of the data throughout the research process. The data were collected during an in-depth immersion in a software organization, and consists of participant observations and organizational texts.

4.1 Philosophical perspectives

In philosophy of knowledge, philosophical assumptions are abstract principles combining beliefs about ontology, epistemology, and methodology. These assumptions shape how researchers see the world and act in it, and provide criteria for evaluating the knowledge they produce (Denzin and Lincoln, 1994; Pfeffer, 1993).

Paradigms (or research paradigms) are formed by the adoption of particular ontological, epistemological, and methodological beliefs. Acknowledging that several paradigms are available to researchers, this section contrasts the philosophical assumptions of the two main research paradigms found in IS research, positivism and interpretivism.

4.1.1 Positivism

Positivist studies are based on the ontological assumption that reality is objective and singular, and independent from the researcher. In this frame, the world is premised on the existence of hard facts that can be measured by adhering to strict rules of procedure (Benbasat et al., 1987). Knowledge typically consists of verified hypotheses which can be accepted as facts (Guba and Lincoln, 1994). For IS researchers adhering to a positivist paradigm, the aim of the inquiry is to achieve explanation, prediction, and control of the phenomena in which IT plays a central role (Orlikowski and Baroudi, 1991).

Owing to its epistemological posture, positivism assumes that researchers can achieve a state of objective detachment from the world in which they are situated. This enlightened state of objectivity makes them able to discover "how things really are" and "how things really work" (Guba and Lincoln, 1994: 108). Thus, within positivism, it is posited that researchers are capable of studying a phenomena without influencing it or being influenced by it (Darke et al., 1998). Researchers following the same procedure for collecting evidence and analyzing data should arrive at the same conclusions in different contexts or at different times. Although qualitative research methods are sometimes adopted in positivist studies, the methods of the natural sciences have traditionally been favoured because of the importance attributed to reliability (Klein and Myers, 1999). Thus, in sum, positivist studies favour an objectivist understanding of socio-technical phenomena-- that is to say, an understanding that breaks the immediate experience of the social world.

Positivism faces significant difficulties when it comes to generating a theory about an organization-specific phenomenon or obtaining insights into the social aspects of IS (Orlikowski and Baroudi, 1991). For a start, positivist research methodologies are ill-equipped to appreciate the richness of the context. In order to control particular variables, these methodologies have to rule out other variables which exist in the context, variables which may have a bearing upon the findings if they are allowed to exert their effects. Moreover, insights into human affairs can hardly be reduced to a few variables without losing their richness and significance (Guba and Lincoln, 1994). Finally, the neutrality of the researcher is, at best, questionable when it comes to examining a complex social phenomenon such as that being considered in this doctoral study.

Because of the problems inherent in positivism, interpretivism has become recognized as a valid alternative paradigm in organization studies in general, and IS research in particular. As Walsham (1995) notes, interpretivism is now part and parcel of mainstream IS research, and the choice between interpretivism and positivism is, more than ever, an important issue for IS researchers.

4.1.2 Interpretivism

Interpretivism is premised on the ontological belief that reality is socially constructed through interactions. IS researchers following this paradigm maintain that a certain technology or state of affairs can have different meanings for informants and researchers (Lee, 1991). Whereas positivist methods tend to bracket some of the particularities of the social-organizational context, interpretivist methods are very much context-sensitive. Generalization of the research results from the setting to a population is generally not sought or believed to be possible (Lee and Baskerville, 2003).

Epistemologically, scientific knowledge is created in interactions between the researchers and the informants. The mathematical logic of natural science is rejected in favour of methods which enable the researcher to understand symbolic actions and subjective meanings (Orlikowski and Baroudi, 1991). It is perhaps in its commitment to the study of the world from the point of view of the interacting individual that interpretivism most stands out (Denzin and Lincoln, 1994). As a result, interpretivism is often equated in the IS literature with the subjectivist pursuit of individual meanings (Silverman, 1998).

In terms of limitations, the interpretivist paradigm offers little internal validity. A research study has internal validity if the variables representing a phenomenon can be accurately measured, controlled, or manipulated. Under the interpretivist paradigm, reality is always multiple or, if what Orlikowski and Baroudi (1991) call a "radical interpretivist" stance is adopted, literally created by the researcher (Latour and Woolgar, 1986). In a similar vein, the extent to which findings can be generalized to other similar settings is always questionable (Lee and Baskerville, 2003). However, it is perhaps on the question of reliability, the extent to which findings can be replicated or reproduced by other scientists, that interpretivist IS studies are the most vehemently criticized (Benbasat et al., 1987; Darke et al., 1998; Walsham, 1993). Finally, the extent to which the findings of interpretive IS studies are free from bias is always a concern. This fact is in evidence in the effort of researchers to triangulate their methods in order to inject objectivity and credibility into the findings (e.g., King, 1996). Table 3 summarizes the philosophical assumptions of the two research paradigms.

Table 3: Two main paradigms in IS research

Philosophical Assumptions	Positivism	Interpretivism
Ontology: What is nature of reality?	Social reality has an existence which is as hard and concrete as the natural world.	Social reality is symbolically constructed and reconstructed by humans.
Epistemology: What is the nature of the relationship between the IS researcher and the known?	The IS researcher can study the organization without influencing it or being influenced by it.	Scientific knowledge is created through interaction among the IS researchers and the informants.
Methodology: How is knowledge of the world gained?	Favors experimental and manipulative methodology based on verification of hypotheses.	Places considerable stress upon exploring the particularities of context and getting close to the informant.

4.2 Some difficulties with subjectivism

The previous description of positivism and interpretivism indicates that these two paradigms sustain different ways of studying a phenomenon. The description highlights the objectivist and subjectivist orientation that positivism and interpretivism respectively favour. To reiterate, objectivism presupposes a break with the immediate experience. This orientation seeks to understand the immutable principles governing the social world without necessitating the researcher to become part of it. On the other hand, subjectivism seeks to grasp the way the social world appears to the individuals who are situated within it. Subjectivism presupposes the possibility of understanding the lived experience of others, and theorizes that such an understanding can, by itself, derive an adequate form of knowledge about the social world (Thompson, 1991: 11).

The objective of this doctoral research, as the research questions formulated in Chapter 2 make clear, is to increase our understanding of a social phenomenon within a specific cultural and contextual setting. Particular attention is paid to the context within which beliefs about software development and software development

practices are established as legitimate. The research does not seek to measure variables, test hypotheses, or produce generalizable knowledge. Rather, it seeks to develop a meaningful and theoretically rigorous account of a phenomenon by utilizing theories so far little-known to IS research. In this sense, the research adheres predominantly to the philosophical assumptions underlying the interpretive paradigm (Cavaye, 1996).

However, for reasons that will be justified as this chapter progresses, the researcher endeavors to move beyond the subjectivism characteristically associated with the interpretive paradigm, yet to avoid relapsing into the objectivism of the positivist paradigm. In order to identify some misconceptions about the interpretive paradigm and some limitations of subjectivism, the three following claims, contained in IS research, are examined:

1. Interpretive research involves an enquiry from the perspective of the informants;
2. Subjective understanding is the de facto alternative to objective understanding;
3. A subjectivist understanding of the social word is trustworthy.

Claim 1: Interpretivism as a subjectivist approach

It is widely assumed in the IS literature that interpretive research implies enquiry from the point of view of the participants. For example, Orlikowski and Baroudi (1991) write that "The primary endeavor [of the interpretive research approach] is to describe, interpret, analyze, and understand the social world from the participant's perspective…" and that interpretive "researchers thus attempt to understand phenomena through assessing the meanings that participants assign to them."

In the same vein, Walsham (1995) contends that to be interpretive, IS research must provide "evidence of a nondeterministic perspective, and intent to increase understanding of the phenomena within a specific cultural and contextual setting, and an examination of the phenomena and the setting *from the perspective of the participant*" (emphasis mine). Moreover, Walsham implies that IS research which does not adhere to these three criteria simultaneously is not interpretive research.

Claim 2: Objectivism and subjectivism are the two alternatives

In the IS literature, the interpretive research approach is generally seen as the alternative to the positivist research approach. As Lee (1991) remarks, "it often appears that the two approaches are opposed," and that the interpretive approach is "the alternative to the positivist approach." Similarly, Walsham (1995) observes that "the epistemological choice between interpretivism and positivism is an important issue for IS researchers." As a consequence, it is generally believed that researchers have to choose between understanding the social world as it appears to the individuals who are situated within it (i.e. the subjectivist orientation) or breaking with the immediate experience in order to identify the principles governing the social world (i.e. the objectivist orientation).[6]

Claim 3: A subjectivist understanding is trustworthy

The third claim asserts that researchers can acquire a trustworthy understanding of the social world by studying how the informants perceive and describe it. However, as indicated by Van Maanen (1979: 546), researchers "can be misled because informants are sometimes totally unaware of certain aspects underlying many of their own activities. Like fish who are presumably unaware of the water in which they swim," informants take the things that are associated with their daily work for granted. Thus, it would not always be useful for researchers to attempt to capture how informants 'see things.' The value of a subjectivist understanding depends largely on the research objective. For example, a study that aims to understand how individuals in a particular context perceive something or themselves may benefit from a subjectivist understanding. However, if the objective is to understand the mechanisms that prevent individuals from conceiving of something in a different way and reject their identity, a purely subjectivist understanding is at best limited because individuals are not likely to be aware of how these mechanisms operate.

The problems associated with the trustworthiness of subjectivism have puzzled social scientists for at least a century. Karl Marx, for example, recognized that any attempt

[6] Although some scholars (Chua, 1986; Orlikowski and Baroudi, 1991) treat critical research as a distinct research paradigm, it is here considered to be a philosophical orientation that can be adopted as part of the positivist or interpretivist paradigm.

to view reality as perceived by the actor would be likely to reproduce the false or twisted version of ideas propagated by the reigning orthodoxy (Marx, 1867). This doctoral dissertation seeks to avoid the pitfalls of relying on a subjectivist approach, while remaining sensitive to the particularities of the context.

4.3 Interpretivism as a preferred research approach

There is no reason to assume that an interpretivist study offers an understanding based only on the perspective of the informants. Equating the subjectivist approach with interpretivism seems to be based upon the misconception that positivist research is 'objective,' in the sense of not being influenced by interpretations or prejudices. Following this erroneous line of thinking, a study that is not 'objective' is 'subjective' and, hence, subjectivist in orientation (Silverman, 1998).

As to what concerns the assumption that objectivism and subjectivism are alternative to one another, it is clear that there exist other possibilities beyond them. It is entirely possible that a researcher is neither seeking to see through the eyes of informants, nor seeking to measure variables or test hypotheses. Suchman (1987), in her influential study of human-computer interaction, offers an example. She focuses on the practices of actors. In so doing, Suchman avoids losing sight of the phenomena by reducing social reality to a set of variables or the definition of the participants.

A practical solution to the limitations of the subjectivist orientation is to focus on what people are doing. Maynard explains how the traps of subjectivism can be sidestepped:

> The question that ethnographers have traditionally asked – "How do participants see things?" – has meant in practice the presumption that reality lies outside the words spoken in a particular time and place. The … [alternative] question – "How do participants do things?" suggests that the microsocial order can be appreciated more fully by studying how speech and other face-to-face behaviours constitute reality within actual, mundane situations. (Maynard, 1989)

Referring to the point made by Maynard above, and the work of ethnographers working within the IS field (Orlikowski, 1991; Suchman, 1987), Silverman (1998)

notes, significantly, "that the simplistic opposition between positivist and interpretivist models of social research is an unhelpful basis for qualitative or case study research." The objective of the research should determine the paradigm adopted and the research strategy to be deployed to fulfil this objective.

The objective of this doctoral research, as the research questions formulated in Chapter 2 make clear, is to increase our understanding of a social phenomenon within a specific cultural and contextual setting. The research does not seek to measure variables, test hypothesis, or draw inferences about phenomena from a typical organization to similar organizations. In this sense, this research adheres predominantly to the philosophical assumptions underlying the interpretive paradigm. However, the identification of interpretivism with an analysis of how informants 'see things' is rejected (Silverman, 1998). An analysis of how people 'do things' is instead chosen. It is believed that by adopting a practice perspective, the risks of being misled by informants' views can be partly avoided. More importantly, it is believed that informants' accounts could not adequately explain how the process of institutionalization of software development practices and beliefs occur in the study organization.

4.4 Research strategy

A research strategy is "a way of going about one's research, embodying a particular style and employing different methods" (Galliers, 1992). The case research strategy involves gathering detailed information about an organization or a functional department (Denzin and Lincoln, 1994; Yin, 1981), and typically combines multiple data collection methods for gathering qualitative and/or quantitative evidence (Eisenhardt, 1989). The case research strategy is utilized for the present doctoral research to gain in-depth knowledge of the process by which beliefs about software development and software development practices come to be established as legitimate within an IT organization. The case study research strategy is appropriate for three principal reasons.

First, as indicated by Benbasat et al. (1987), a case strategy is an appropriate way to research an area in which few previous studies have been carried out. An abundance

of academic literature has been produced on the management of the software process. However, how software development practices come to be deemed legitimate in organizations still puzzles researchers (Madsen et al., 2006; Truex et al., 2000). Positivist research strategies are, therefore, inadequate at this stage because of the great number of variables potentially relevant, but still unknown (Denzin and Lincoln, 1994; Yin, 1981).

Secondly, a study of the process by which a collective agreement about the nature of an object is established has to be conducted in a real-life context. Consequently, a study of the social phenomenon of interest should be conducted in an organizational setting. Russo and Stolterman (2000a) claim that the bias towards positivist research strategies such as surveys "has limited the knowledge of what is actually happening in IS practice." The case study strategy, which involves the study of a phenomenon within its real-life context, offers the possibility of delving into the complexity of the context (Benbasat et al., 1987; Eisenhardt, 1989).

Finally, and closely related to the two previous points, a research strategy that enables the gathering of qualitative data is needed. Although the case research strategy can be used to obtain quantitative data (Lee, 1991), it allows, in the present case, for a combination of two qualitative data collection methods. The idea is to accurately represent what is really happening in the situation, in all its richness and complexity (Benbasat et al., 1987; Yin, 1984).

4.4.1 Other research strategies considered

In addition to the case study, two other research strategies were considered: action research and ethnographic research. Those are action research and ethnographic research. Action research pursues the dual objectives of bringing about positive change in an organization and monitoring the results. According to Rapoport (1970), this research strategy "aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework." Although there is still considerable debate in the IS literature as to the scientific merits of action research

(Baskerville and Myers, 2004; Baskerville and Wood-Harper, 1996), it is widely used with the field because of the close link between the theory and the practice.

Although the researcher was paid for his professional services by the study organization, the present study does not qualify as an action research for three reasons. First, the researcher did not play a role in the organization that was significant enough to instil change (Baskerville and Myers, 2004). Instead of being responsible for the outcome of a specific project, the researcher was involved in several different projects assigned to him by his then-superior. Secondly, to qualify as an 'action research,' members of the organization must become involved in the research at all levels, from defining the aim of the study to writing the results (Robson, 1993). However, the study organization was primarily interested in making use of the professional expertise of the researcher, and not in his research project. Finally, researchers adopting action research as a research strategy should tell practitioners of the host organization something useful, or at least attempt to do so. This research was not conducted with this aim in mind. However, the researcher hopes that the conclusions reached in this study will eventually inform the work of IS practitioners.

Ethnographic research is the other research strategy that was considered by the researcher. The distinction between in-depth case studies and ethnographies is often blurred (Klein and Myers, 1999). Ethnographies require the researcher to spend a considerable amount of time in the field. It is not uncommon that researchers spend more than a year in the field in order to place the phenomenon studied in its social and cultural context (Reeves Saunday, 1979). Although ten months were spent in the field and a thorough understanding of the social and cultural context was sought, the researcher did not pursue "the essential ethnographic question" of what it is to be a member of the organization (Van Maanen, 1988). The researcher did not "go native" and always saw himself as a researcher having infiltrated the corporation for the purpose of carrying out a study. Ethnographic data collection methods, such as recording observations in a diary, were, nonetheless, used.

4.5 Data collection method

In interpretive research, the most commonly-used data collection methods are interview methods and observation methods (Creswell, 1998; Denzin and Lincoln, 1994). Interview methods differ from observation methods in that they are interventionist – they interfere to some degree with the natural stream of everyday life. Researchers using interview methods often ask the informants questions, pose tasks to them, or deliberately confront them by pointing out the contradictions present in their accounts. Interviews are often conducted outside the normal flow of activities. Researchers relying on observation methods, on the other hand, seek to gather impressions of the surrounding world and witness the phenomena they are studying in action. This task usually requires direct contact with the observed individuals as they go about their routine daily activities (Adler and Adler, 1994). For this reasons, observation methods are particularly well-suited to a study of everyday practices.

Researchers utilizing observation methods can take roles that range from hidden voyeur, who watches from outside, to active participant. Some advantages of opting for participant observation include direct exposure to the social context and to the flow of interaction among informants. Thus, even if an understanding from the point of view of the informant is not sought, participant observation enables researchers to take part in organizational activities to gather evidence without disrupting the natural stream of everyday organizational life (Adler and Adler, 1994: 378). Participant observation has the potential to generate findings that more accurately represent what is really happening in the situation than interview methods.

In the previous chapter of this doctoral dissertation, a discursive theoretical approach was proposed to illuminate the research questions previously posed. Organizational discourse theory was presented as a potent theory for understanding how beliefs are negotiated and established in an organization, and how such struggles shape organizational practices. A key idea underlying the theoretical framework is that the analysis of discourses should not be isolated from an analysis of the socio-institutional contexts in which discourses are embedded. This is because the context is deemed to influence the value that discourses are endowed with-- that is, the manner in which organizational actors normally "valorize" (i.e. assess the value of)

discourses (Bourdieu, 1991; Thompson, 1990). In order to address the particularity of the context, a context-sensitive theoretical framework was favoured over conversation analysis and other such text-centric theories. In the light of these research objectives, participant observation emerged as an appropriate data collection method.

4.5.1 Participant observation

Participant observation was utilized as a data collection method in order to obtain a rich understanding of the context. Over a ten-month period, from March 2003 to January 2004, the researcher spent between 40 and 50 hours per week in the organization. During this period, the researcher served as an employee with the organization's governance group. Organizational members were informed that the researcher was doing a study, but did not show any interest in it. Probably because of his status as employee and his professional background, organizational members identified the researcher, from his first day in the workplace, as being one of them. Thus, the presence of the researcher did not represent an intrusion in the corporate workers' natural stream of everyday life.

An electronic diary was meticulously kept. In this diary, the researcher recorded observations about a wide range of practices enacted by organizational members as part of their routine daily activities. Observations were recorded as discreetly as possible, typically after work hours. However, any relevant informal verbatim statements that were formulated by organizational actors were noted as soon as possible after their occurrence.

The diary consisted of Microsoft Word documents, usually created for a specific workday. By organizing the diary chronologically, rather than thematically, the researcher could easily go back to previous pages of the diary and further develop a previously-recorded observation from a different point of view, or in the light of a new observation. About one hundred and fifty files were created, resulting in a diary of more than one hundred and forty pages.

Participant observations provided rich insights into the sort of ideas and values that were esteemed in the organization. For example, the care that organizational actors took in cultivating a professional image and the importance granted to producing written documents were identified as a particularity of the study organization. The observations recorded were often juxtaposed with some organizational text collected.

4.5.2 Formal documents collection

Formal documents that described the vision, mission and business strategy of the study organization were collected and examined. These documents were typically electronic files that were discovered in the numerous Lotus Notes databases and on intranets of the organization. The documents included software development codes of practice, presentation material, quarterly organizational objectives, software development standards, and other such organizationally-sanctioned documents. It is believed that these documents were carefully-engineered texts that had been designed to circulate ideas which reflected the reigning orthodoxy. Forty formal documents were collected in total.

The researcher realized during the second month of immersion that formal documents said little about the process by which beliefs about software development and software development practices were established as legitimate. In retrospect, informal documents provided more valid data and proved to be more analytically meaningful than formal documents.

4.5.3 Informal documents collection

Informal documents differ from formal documents in that they are not intended to be distributed widely across the organization. In the context of the immersion, these documents included, but were not limited to, minutes of meetings, memoranda, and emails. Informal documents were encountered by the researcher during his day-to-day work and were also found in the Lotus Notes databases of the firm.

These organizational texts provided the researcher with a window on the practical problems associated with the orderly, and sometimes naïve, perspective offered by

formal documents. Informal documents often expressed alternative ideas and provided insights into how the legitimacy of some beliefs had been established. Furthermore, studying these documents enabled the researcher to examine the evolution of some ideas before beginning the study. For instance, it is by scrutinizing informal documents that the researcher made sense of how some organizational initiatives intended to standardize software development had come about. In total, more than five hundred and sixty informal documents were collected.

4.5.4 Interviews

In January 2004, during the last two weeks of fieldwork, the researcher conducted twenty five interviews in order to clarify some ambiguities. One question the researcher needed to further explore was the extent to which organizational members were conscious of their actions. Did they do what they did and say what they said with a conscious intent in mind, or did they function instinctively? It will become clear as this dissertation progresses that understanding the extent to which the context structures the discursive and non-discursive practices of organizational actors is analytically vital.

Twenty two individuals of diverse hierarchical levels and backgrounds were interviewed. The interviews lasted, on average, fifty minutes and were conducted outside the normal work hours. All interviews were tape-recorded. The researcher did not judge it necessary to transcribe the interviews because interview data are not a source of findings for this study. Interviews were conducted in order to clarify issues, rather than to generate findings.

4.6 Data interpretation

All data obtained were qualitative. The literature states that the methods used to interpret qualitative data can be divided into two categories: quantifying methods and non-quantifying methods (Hussey and Hussey, 1997: 248-249). Quantifying methods involve turning the qualitative data into a numerical data. However, because of the ontology favoured, a non-quantifying method was deemed more desirable to preserving as much as possible the deep meaning of the qualitative data (Kaplan and

Maxwell, 1994). Thematic classification (i.e. coding) was, nonetheless, used as a means of synthesizing, referencing categories, and retrieving data.

The organizational texts collected and analyzed were classified by themes. Proceeding inductively (Miles and Huberman, 1994), the researcher did not begin the analysis of data with a set of themes in mind, but rather let these themes emerge. Themes were progressively delineated as organizational texts were iteratively re-examined. It was not uncommon that organizational texts superimposed, knotted into one another and were found to belong to more than one theme. In keeping with the discourse analytic approach adopted, those themes are considered to be organizational discourses in their own right-- that is to say, sets of statements that bring organizationally-related objects into being (Grant and Hardy, 2003: 6). In order to reduce the biases inherent in the researcher's interpretation of the texts, a form of hermeneutics was utilized (see below).

4.6.1 How discourse analysis is done

A large number of non-quantifying methods can be used to reveal the discourses inhabiting organizational texts. The researcher did not feel the need to utilize a well-known method, and broadly followed some of the principles for analyzing discourses laid out by Parker (1992). In broad strokes, the analysis of texts and discourses involved three phases: identifying discourses, examining the points of tension within discourses, and discovering the pattern of relationship among discourses. Figure 1 provides a brief description of these phases, as well as identifies the key questions that need to be addressed phase 1 and phase 2.

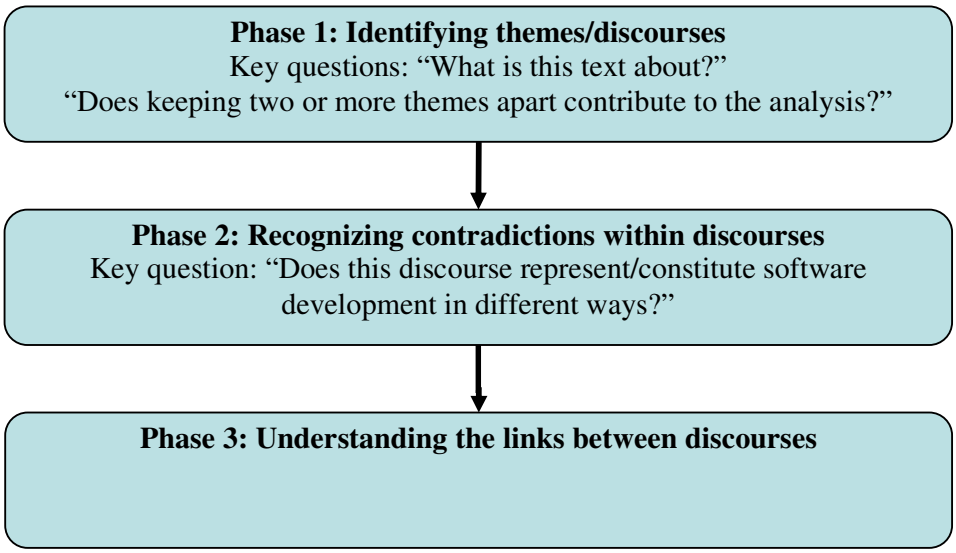


Figure 1: The analysis of texts and discourses

In the first phase of analysis, general themes present in organizational texts were identified by asking and answering the question, "What is this text about?" The following excerpt from an internal report about software process improvement is helpful for illustrating the process of discourse identification:

> […] we can approximate a statistical quality control over the human failures that plague projects: underestimation, schedule slippage, requirements mismatch, and so on. We must begin by cataloguing such failures and learning from their patterns. The PIR plays a key role in this process. [Internal report, 01-08-2003]

In the previous excerpt, the acronym "PIR" refers to a project retrospective. In software engineering, a project retrospective is a process carried out by a project team at the end of a project, aimed at discussing what was successful about the project covered by that retrospective, what could be improved, and how to the successes and improvements of the project could be incorporated into future projects. The excerpt justifies the rationale for systematically completing the PIR. It is about the PIR. Moreover, the text argues that the PIR has to involve statistical quality control. Thus, the text is also about 'measurement.' In a similar vein, the researcher interprets issues of estimation and schedule slippage as difficulties associated with

predicting. Thus, the text segment, "human failures that plague projects: underestimation, schedule slippage," refers to the theme, 'predictability.' Finally, the text segment, "learning from their patterns," refers to the theme, 'learning.'

Being familiar with the texts collected was necessary to identifying the relevant themes. For example, the researcher realized through an ongoing analysis of the findings that although the word 'quality' is used in the previous extract, 'quality,' cannot be considered a theme because it is too all-encompassing. If quality is broadly understood to be the ability of a software application or component to fulfil the requirements of the customer (IEEE, 1990), then everything that people do in a software organization has to do, in one way or another, with producing or managing quality.

On the other hand, 'statistical quality control' was, at some point in the data interpretation process, aggregated with other themes to become 'measurement' because this theme offered a coherent set of statements referring to software development. Keeping the themes apart would not have contributed to the analysis, while further aggregating 'measurement' with other themes would have hampered the analysis. Seven themes were, in total, identified. These themes sustain certain visions of software development and, hence, are considered to be organizational discourses.

In a second phase of analysis, following the principles for analysis discourse laid out by Parker (1992), the researcher examined the contradictions that existed within discourses by asking and answering the question, "Does the discourse represent or constitute software development in different ways?" In the foregoing extract, for example, 'learning' is associated with the process of systematically identifying and measuring unwanted patterns. This perspective on learning stands in marked contrast to that promoted by agile development. For advocates of agile development, as will be explained in detail in this doctoral dissertation, learning occurs at a human level, rather than at a process level. Thus, 'learning' is found to carry two contradictory meanings.

Finally, still following Parker's (1992) principles, the researcher determined how the seven organizational discourses identified related to, or opposed, one another. The researcher found hand-drawn diagrams and tables very useful tools. By going back to the data and by using mapping techniques, the researcher could see how some discourses depicted software development coherently. For example, in the previous extract, 'measurement' and 'learning' were identified as mutually supportive discourses: measuring enables learning.

Ricoeur's hermeneutics of suspicion (Ricoeur, 1974; 1981; 1991), which involves formulating and comparing different interpretations of a text, proved particularly useful for identifying contradictions within discourses. The next section explains what the hermeneutics of suspicions involves and how it informed the interpretation of organizational texts.

4.7 Hermeneutics

Hermeneutics is generally described as a mode of analysis (or a philosophy) that strives to recreate or re-experience the thoughts of the author of a text in order to allow for a better understanding of what the text means (e.g., Kets de Vries and Miller, 1987). While this concern to 'view through the eyes' of the author is central to many forms of hermeneutics, it is secondary in Paul Ricoeur's work (Ricoeur, 1981; 1991). This section explains how the epistemology underlying Ricoeur's hermeneutic of suspicion was adopted as part of the analysis of organizational texts and how it enriched the analysis.

It is well-known to discourse analysts that texts, especially texts produced in informal oral communication, often take sudden shifts, leave ideas incomplete, and are contradictory (Fairclough and Wodak, 1997). Consequently, in most cases, analysts have to use their discretion when interpreting texts. Unquestionably, analysts want to understand texts in all their complexity and to grasp the objectives which have motivated text producers. In practice, however, this goal is never possible because text producers and analysts are different entities. Text producers and analysts are often situated in different social contexts, have different backgrounds, and pursue different objectives.

Ricoeur's hermeneutics of suspicion is a paradigm of text interpretation developed to interpret the texts that constitute discourses. Central to this paradigm is the notion that texts take an autonomous character once produced. Seen in this light, texts do not contain fixed meanings that can allegedly be recovered through a socio-historical reconstruction. Rather, texts invite plural readings and interpretations. The role of the analyst, who is presumed incapable of recovering the original meaning of texts, is to distil from texts an interpretation that opens a window of understanding.

The hermeneutic of suspicion actively seeks to overcome the effects of ideologies. For Ricoeur, text interpretation becomes ideologically distorted when one transposes one's ways of thinking or points of view on the texts. However, Ricoeur is wary of other forms of hermeneutic that seek to project the analyst in a context that is not his. Whereas Gadamer's philosophical hermeneutic (Gadamer, 1977) and Habermas' critical hermeneutics (Habermas, 1980) both strive to recreate the socio-historical conditions of the text producer, the hermeneutics of suspicion seeks to strip away layers of ideological historical and personal distortions. Ricoeur does not, however, claim that discourse analysts can arrive at an ideology-free, authentic account through a hermeneutic mode of analysis. Because "the critique of ideology is a task which must always be begun, but which in principle can never be completed" (Ricoeur, 1981: 245), the objective of the hermeneutic is to develop interpretations which will contribute to understanding the texts without denying the possibility of alternative interpretations.

For the purpose of this doctoral research, the researcher adopted the epistemology underlying the hermeneutic of suspicion to interpret the organizational texts collected and to identify the discourses that inhabit them. In simple terms, this meant acknowledging that a particular organizational text can be understood differently, and making an effort to develop alternative interpretations. In doing so, the researcher sought to go beyond what appeared immediately obvious to him in the organizational texts. In detailed terms, employing the hermeneutics involved (a) carrying out cross-checking of interpretations through iterative hermeneutical circles; (b) relating single texts to the organization's intertextual space within every iterative hermeneutical circles; and (c) acknowledging that a "true" interpretation cannot be reached.

It was found that employing the hermeneutic of suspicion injected rigor into the analysis of organizational texts. Employing this paradigm of text interpretation proved particularly rewarding in the third phase of analysis described above (see Figure 1). It helped the researcher to recognize the contradictory ways in which some organizational discourses could be understood-- for example, the different meanings 'learning' can be endowed with (see above). Moreover, although hermeneutics is not unknown to the field of IS research (Boland, 1991; Lee, 1994; Myers, 1995), the explicit application of the hermeneutic of suspicion represents an addition to IS research because the principle of suspicion is by far the least-developed in the IS research literature (Klein and Myers, 1999). The researcher thus gained first-hand experience in a means of engaging with organizational texts that is little-known in the field of IS research.

5 Case Description

John Pierport Morgan (1837-1913) is considered by many the greatest financier in the history of United States business. Morgan joined his father's banking firm in 1856 and established J.P. Morgan and Co. (hereinafter J.P. Morgan) in 1895, which became specialized in financing American business and marketing American securities to Europe. In the aftermath of the Civil War (1861-1865), J.P. Morgan helped pay off the country's enormous debt and raised vast amounts of money in foreign investment to help build railroads in the United States

As the most prominent financial service company of the Progressive Era (1900-1920), J.P. Morgan financed many of the giants of the America's industrial heyday. Often, it entered into the councils of these companies and acquired control over their administrations. The firm also exerted considerable power outside the United States, endowing its actions with broad significance in terms of American foreign policies. J.P. Morgan is reported to have stopped panics, saved the gold standard, rescued New York City from bankruptcy, and, less famously, coaxed America into war for profit (Chernow, 1986) three times.

The Chase Manhattan, although also associated with the leading figures of corporate America, and bigger than J.P. Morgan in terms of assets, has never enjoyed the patriotic reputation J.P. Morgan possessed. This, however, did not prevent Chase Manhattan from buying J.P. Morgan in a stock deal in September, 2000. The acquisition was a means for the Chase Manhattan, a commercial bank, to take a big step forward in investment banking following the dismantlement of the Glass-Steagall Act, and to acquire the prestigious J.P. Morgan name.[7] The new company was called J.P. Morgan Chase and Company (hereinafter J.P. Morgan Chase), combining two of the most established names in U.S. banking. Three years later, in

[7] As a consequence of the bank runs and failures of the Great Depression, Congress passed Glass-Steagall Act in 1933 in order to protected bank depositors from the additional risks associated with security transactions. The act prohibited commercial banks offering general services to businesses from collaborating with investment banks issuing stocks and bonds. J.P. Morgan chose to remain an investment bank. In 1999, the Glass-Steagall Act was dismantled as a result of the Financial Services Modernization Act of 1998.

July 2004, the newly-formed company acquired Bank One Corporation, bringing its total assets to $1.1 trillion. Table 4 presents the assets (in billions of US dollars) of the major American banks prior to the acquisition of J.P. Morgan by Chase Manhattan.

Table 4: Pre-acquisition assets of America's biggest banks

Bank	Assets ($ billion) as of Quarter 2 of 2000
Citigroup	$792
Bank of America	$680
Chase Manhattan	$396
Bank One	$273
J.P. Morgan	$266
First Union	$258
Wells Fargo	$234
Washington Mutual	$186
FleetBoston Financial	$181
SunTrust	$100

Source: AmericanBanker.com, accessed 12 January 2005

5.1.1 Mission and structure

J.P. Morgan Chase operates six divisions: asset and wealth management, card services, commercial banking, investment banking, retail financial services, and treasury and securities services. The study was conducted within the investment banking division of the firm. Investment Bank Technology (IBTech) is the IT organization responsible for developing and maintaining the technological infrastructure and software products of the investment banking division. IBTech's mission is to "deliver innovative and commercial solutions in close partnership with the business and operations through a committed team of outstanding technologists" [A27]. The organization has approximately 830 permanent employees and 200 contractors in 10 locations. Table 5 presents a breakdown of the number of permanent employee per location.

At the time the study was carried out, IBTech had ten major IT projects in progress. These projects were intended to enable the business growth of the investment banking division by $917 million, and to generate efficiency and productivity savings of $92 million over a five-year period (2003 to 2007). The total technology

spending was budgeted at $122 million for this period, of which approximately one third was spent while fieldwork was being conducted.

Table 5: Number of permanent employees at IBTech per locations

City	Number of permanent employees
London	386
New York	217
Tokyo	52
Hong Kong	40
Dover (Delaware, USA)	37
Glasgow	35
Sydney	32
Mumbai	12
Johannesburg	12
Singapore	10

The IT products developed at IBTech are, in most cases, highly innovative. They include, among others, a proprietary global equity trading platform, a novel settlement and clearing infrastructure for the EMEA region (Europe, Middle East and Africa), and an advanced revenue-reporting engine to calculate commissions on transactions.

Appendix 1 shows that the organization structure adopted by IBTech was rather conventional. Being part of one of the world's largest and most established banks gave IBTech poise, poise reflected in the adoption of a traditional hierarchical organization structure to ensure coordination. Chief Business Technologists are responsible for delivering information systems in a region (i.e., EMEA, Asia, Americas) or for a global line of business (i.e., Futures and Options, Equities Research). Global Managers are accountable for standardizing and making the work of others more efficient by managing global programs. Chief Business Technologists and Global Managers report directly to IBTech's Chief Technology Officer.

5.1.2 IBTech as an innovative high-tech organization

IBTech differs substantially from the archetypical innovative organization depicted in popular press and textbooks. Mintzberg's conceptualization of the Adhocracy

provides a useful starting point for discussing some of the features of innovative organizations:

> To innovate means to break away from established patterns. So the innovative organization cannot rely on any form of standardization for coordination. In other words, it must avoid all the trappings of bureaucratic structure, notably sharp divisions of labor, extensive unit differentiation, highly formalized behaviors, and an emphasis on planning and control systems (Mintzberg, 1979: 432-433).

Regarding the development and assimilation of standardized practices, Mintzberg remarks that "the Adhocracy cannot rely on the standardized skills of these experts to achieve coordination, because that would lead to standardization instead of innovation" (Mintzberg, 1979: 434). Similar portrayals of the innovative organization can also be found in the writings of Kanter (1988), Quinn (1985), Tushman & Nadler (1986), and Van de Ven (1986), among others.

Whether developed internally or acquired from vendors, standardized practices play, at IBTech, a central role in enabling new product development. Perhaps because of its sector of activity, its history, or diverse contingency factors (Mintzberg, 1983; Pettigrew, 1979), the organization is highly bureaucratic and reliant on the formalization of behaviors. However, in spite of these fundamental differences from what is expected of a 'typical' innovative organization, IBTech can clearly be characterized as such.

IBTech's IT products are highly knowledge-intensive and characterized by a high degree of novelty. Products developed by the organization are, in general, distinctive from any other products which exist in the competitive environment. They are typically developed to meet specific needs and usually help the investment bank division to perform mission-critical activities more efficiently, with less risk, and/or at lower costs, than its competitors. Although statistical information about the profile of the workforce could not be obtained for this study, it is obvious that it is predominantly composed of corporate workers whose knowledge and skills grant them the ability to directly contribute to the design and development of technological products.

IBTech operates more than sixty proprietary information systems [A40].[8] These systems include sophisticated order and executions trading systems, risk management applications, and communication platforms linking multiple exchanges. In many cases, the information systems are used by a broad institutional client base, ranging from financial institutions, asset managers and industrial groups to professional traders and private clients, and represent important sources of revenues for J.P. Morgan Chase. Needless to say, the knowledge and creativity embedded in the IS is fiercely protected by an array of patents.

IBTech supports many programs designed to foster innovation. Within the past two years, through the 'IB Technology Patent Program,' employees have obtained twelve patents on IBTech's behalf. Those innovative ideas were either incorporated into existing products or used internally by the employees in their day-to-day activities. The 'Productivity and Innovation Initiative,' another program intended to foster innovation, delivered a novel data storage technique following a research and development project. A third program, 'Business Innovation Through Technology,' seeks to foster a culture of innovation and the transfer of knowledge from adjacent fields. In sum, although IBTech is part of a conservative corporation, it possesses many characteristics of an innovative organization.

At IBTech, it is believed that organizational learning fosters innovation. The 'learning organization' program has as its objective to transform IBTech into a "Learning Organization." An organizational document describing the 'learning organization' program defines a "Learning Organization" in the following terms: "An Organization that learns and encourages learning among its people. It promotes exchange of information between people hence creating a more knowledgeable workforce. This produces a very flexible organization where people will learning from experience and adapt to new ideas and change through a shared vision" [A31].

As will become clear as the discussion proceeds, in the present context, learning centres principally on the development of standardized software development practices and is closely connected to software process improvement. The 'learning

[8] A capital 'A' followed by a number in between square brackets '[ ]' refers to the code given by the researcher to an organizational text.

organization' program is based on six mechanisms. Table 6 presents these mechanisms and specifies their importance in terms of effort and budget.

Table 6: The 'learning organization' program and its mechanisms

Mechanism	Effort (Months)	Budget (US$)
CMM & CMMI: The models consist of best practices that address the development and maintenance of products and services, covering the product lifecycle from conception through delivery and maintenance.	30	$675,000
EPF: The equity process framework (EPF) is a standard software development process. It is necessary for the organization to institutionalize a standard software development process to be accredited CMM level 3.	45	$1,012,500
SEPG: The software engineering process group (SEPG) is the forum intended to facilitate the definition, maintenance, improvement and establishment of the software development processes for IBTech. SEPG allows areas for improvement to be identified and good practices to be retained.	10	$225,000
PIR: The post implementation review (PIR) is a project retrospective used to prioritize process improvement actions to the SEPG.	7.5	$168,750
Metrics: Performance indicators are devised to enable the measurement of improvement. The development of metrics is necessary for the CMM & CMMI initiatives.	2.5	$56,250
Knowledge Management: Training and the development of knowledge management intranet fall under this category.	3	$67,500

Source: [A31]

The 'learning organization' program is an umbrella for closely-related mechanisms intended to create and implement more effective software development practices. This program is comprised of a formal governance structure that ensures that learning opportunities are systematically identified and exploited by the whole organization. Figure 2 illustrates this structure. According to this structure, regional Software Engineering Process Groups (SEPG) are responsible for presenting to the SEPG Steering Committee practices that have been found to provide good results in a particular software house, and that present learning opportunities for other regions. These local 'best practices' are identified by the software houses' quality working

groups. Moreover, several SEPG Working Groups exist at IBTech, and aim to advance knowledge in particular areas, such as agile and iterative development and the use of metrics, for the whole organization. The regional SEPG are established by, and report to, the SEPG Steering Committee. The SEPG Steering Committee is made up of Chief Technology Officer, the Chief Business Technologists, and the leaders of a few quality working groups. It is the existence of a program intended to foster learning, in addition to the existence of this formally-organized group, which leads some corporate workers to describe their organization as a 'learning organization.'

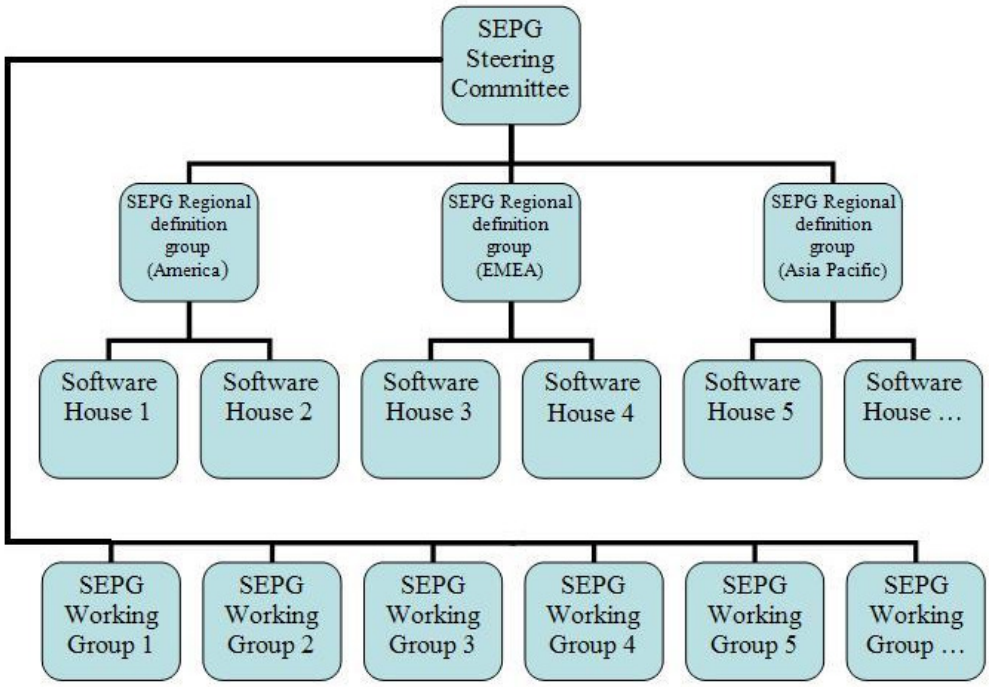


Figure 2: The learning organization governance structure

5.2 The standardization of software development

This section provides an overview of the evolution of the 'learning organization' program. This program was intended to standardize software development practices across IBTech's ten software houses. In a first stage, following the precepts of the CMM/I, IBTech attempted to build its standard software development process from internal best practices. The organization later went on to acquire two commercial methodologies.

5.2.1 The Capability Maturity Model

The CMM is a framework that provides the guidelines for developing a disciplined software process. The original concept of the framework was developed in the early 1980s by Watts Humphrey and his colleagues at IBM. Humphrey's unique insight was that software organizations had to remove impediments to continuous improvement in a specific order if they were to keep improving their processes capability over time. Readers who are not already familiar with the CMM can refer to Appendix 2 for an overview.

The CMM defines five maturity levels, each of which builds on the lower levels by adding additional practices (see Table 7). At Level 1, the software process is ad hoc and chaotic. In order to progress to Level 2, basic project management processes are introduced to track costs and schedule. At Level 3, the software process is documented and standardized across the organization. At Level 4, the software process is quantitatively managed and controlled. Finally, at Level 5, the software process is optimized.

Since its introduction in 1991, the framework has spread across industries and has achieved significant penetration into commercial IT. Since 2001, however, the Capability Maturity Model Integration (CMMI) has progressively replaced the CMM. The CMMI encompasses the CMM, but possesses two additional disciplines: 'supplier evaluation' and 'contract monitoring.' IBTech used the two models, and in order to facilitate further discussion, the term 'CMM/I' will be used in this text to refer to the two models without distinction.

Table 7: The CMM for software

Maturity Level	Generic goals	Practices
5	Institutionalize an optimizing process	Ensure continuous process improvement Correct root causes of problems
4	Institutionalize a quantitatively managed process	Establish quantitative objective for the process Stabilize subprocess performance
3	Institutionalize a defined process	Establish a defined process Collect improvement information
2	Institutionalize a managed process	Establish an organizational policy Plan the process Provide resources Assign responsibility Train people Manage configuration Identify and involve relevant stakeholders Monitor and control the process Objectively evaluate adherence Review status with high-level management
1	Achieve specific goals	Perform base practices

IBTech began to implement process improvement based on the CMM/I with one pilot project in 2001, and became fully compliant with the level 2 requirements at the end of the year 2003. In practical terms, this means that requirements were managed and that processes were planned, performed, measured, and controlled. Corporate workers had demonstrated that projects were performed and managed according to their documented plan, even when they were working under pressure, and that a certain degree of consistency had been achieved across projects. Moreover, corporate workers had demonstrated that stakeholders were involved in reviewing the quality of the applications.

A Maturity Level 2 organization does not have to have a set of standard processes. At the end of the year 2003, when it reached a CMM/I Level 2, IBTech had ten software houses spread across the globe adhering to different processes. A few weeks later, when it revealed its objectives for 2004, the organization set itself the ambitious objective of becoming a Level 3 organization by the end of year.

A critical distinction between Level 2 and Level 3 is the degree of process consistency established across the organization. At Maturity Level 3, the processes are more consistently defined and applied because they are based on standard processes. Processes are well understood and applied, and are explicitly described in a 'process asset library.' Thus, the process asset library stipulates what process can be used. Software houses may, nonetheless, tailor their processes from the standard processes to suit particular needs. However, if a software house considers it appropriate to tailor a process, it must do so in accordance with the tailoring guidelines because the principal objective is to institutionalize standard processes across the organization. Institutionalization implies that the process is ingrained in the way the work is performed within the organization.

Another critical distinction between Level 2 and Level 3 is the emphasis given to measures and other process improvement information that relate to describing, implementing, and improving processes (see Table 7). At Level 3, an organization must have the mechanisms necessary to gather improvement information and enable the continuous improvement of the standard processes. As will be explained in details in this section, a post-implementation review process (PIR) and a software engineering process group (SEPG) were used in the study organization to this achieve this end (see Table 6).

During the fourth quarter of the year 2004, corporate workers involved in the 'learning organization' program were becoming increasingly aware of the challenges posed by institutionalizing a standard software process. It was clear that a group from the head office could not design and impose a standard software process on geographically-dispersed software houses. Corporate workers would simply not abandon their existing practices to absorb a new one overnight, even if they were formally told to do so – the exception being corporate workers based in Asia, who were known at IBTech for their obedience. As a result, a collaborative approach based on trust was adopted. It was claimed that such an approach had been made possible by the cultural change engendered by the 'learning organization' program.

Document evidence collected by the researcher indicates that the architects of the 'learning organization' program liked to emphasize the notion that within one year,

they had achieved a radical 'cultural transformation.' An electronic presentation document asserted that individuals who were, in 2003, initially hostile to the change brought about by the CMM/I, could, only a year later, not get enough of process improvement, thanks to this 'cultural transformation':

> The initial CMM thoughts were "It will add no value to anybody, us or the Business", "CMM is inappropriate for us", "Nice in theory", "Are you a boy scout after another badge!" […]
>
> In February 2004, teams bought-in and committed to the Learning Organization Transformation Program. […] A culture of continuous process improvement is now in place at team level. CMM level 3 is a 2004 objective defined and committed by teams themselves with no push from the management. […] Process improvement is possible at the organizational level and can be driven by individuals from team. [Electronic presentation file, June 2004]

The foregoing excerpt may easily lead one to assume that corporate workers generally embraced the process improvement objective and, indeed, spearheaded it. There is, indeed, no question that individuals attached to the software houses played a central role in defining local best practices and actively worked for their institutionalization. These individuals played this role through their involvement in SEPG, and were members of the software houses' quality working groups. However, contrary to what the text implies, the development of a standard software process was not unanimously embraced or advocated by all. In fact, as far as the CMM/I was concerned, the general feeling was that of mild indifference.

The following excerpt from an email (written at a time IBTech should have been progressing significantly towards achieving its process improvement goal) more accurately reflects the passivity of corporate workers and the key role that individuals involved in the 'learning organization' program played:

> About CMM level 3, we **have** to make some progress by the end of this year. […] As soon as we have CMM L3 success stories in the AD [application development] teams (this year hopefully), I'm confident that we will be able to leverage these success stories to get other believers and buyers. The timeframe is tight but we have no choice. [Head of Governance, email 21-09-2004. Emphasis in original document]

The two previous extracts indicate that corporate workers were well aware that moving from Level 2 to Level 3 involved changing how software development was executed. They were also cognizant that it involved developing a "culture of continuous process improvement," and standardizing how software development was thought of. For this reason, corporate workers recognized the importance of creating what CMM/I authorities call a "shared vision" [A31]-- that is to say, "a common understanding of guiding principles including mission, objectives, expected behavior, values, and final outcomes which are developed and used by a group such as an organization" (Chrissis et al., 2003: 628).

5.2.2 The launch of the GreenBook Transformation program

The idea of transforming IBTech into a learning organization-- a concept which had existed in a latent form for years-- became concrete with the launch of a high-profile program in 2003. The "GreenBook Transformation" program was officially kicked off in May with an exceptionally long newsletter:

> The GreenBook was created and launched four years ago. It has been the starting point for process assessment and improvement. Today, a transformation is required to adapt to our continually changing environment. [Newsletter, May 2003]

For most IBTech employees, it came as a surprise that the GreenBook had been created four years previously. Except for those actively involved in quality working groups, no one had a clear idea of what the GreenBook actually was. Despite this fact, its goal seemed harmless enough:

> The GreenBook program has the goal of transforming IBTech Equities into a "**learning organization**", this being the only way to adapt, survive and succeed in an increasingly distributed, complex environment and a competitive, uncertain world. [Newsletter, May 2003; emphasis in original document]

To clarify things up front, the memo defined a learning organization as an organization skilled at "engaging all employees," "creating, acquiring, retaining, and

transferring knowledge," and "modifying it behavior to assess and improve its products and processes continuously" [Newsletter, May 2003].

The newsletter made clear that the GreenBook had a specific role to play when it came to helping IBTech become a more effective 'learning organization':

> we need to develop and organizational capability that will allow us to:
> - Learn from experience and best practices
> - Systematically solve problems and experiment with new approaches
> - Acquire and transfer knowledge quickly and efficiency throughout the Organization.
>
> The GreenBook Transformation will enable this organizational capability. [Newsletter, May 2003; emphasis in original document]

The second section of the newsletter opened by giving an update on the current status of the initiative:

> During the last two years, multiple Codes of Practice have been created in each region [America, Asia, EMEA] in most cases several per Software House. These COPs make some reference to the GreenBook processes. This has allowed us to achieve a certain maturity level corresponding to the Capability Maturity Model (CMM) level 2. From a management perspective this is a considerable achievement, supporting better visibility and increased control at the Software House level. [Newsletter, May 2003]

It is not clear what the author of the newsletter sought to achieve by saying that the codes of practices developed within software houses made reference to the GreenBook. This was not possible because the GreenBook existed only in an embryonic form at the time the newsletter was published. The existence of documented processes at software-house level was, nonetheless, sufficient for the software houses to achieve a CMM/I level 2:

> The GreenBook is our central body of standards and practices and encompasses IBTech's best practices. It is based on existing codes of practice and leverages local expertise. Having such a body of standards and practices is necessary for IBTech to be accredited CMM level 3. However, the main motivation for developing the GreenBook is not the CMM accreditation, but rather the development and implementation of efficient, rationalized work practices. In doing so, we eliminate

> unnecessary procedural duplication and bureaucracy, which make us more "mature" according to external quality/maturity standard. [Newsletter, May 2003]

In the newsletter, an attempt was made to create a distinction between the CMM and the GreenBook. The CMM was a process maturity framework. It was used at IBTech "to assess our maturity and define our gap against the next level." The GreenBook was the "body of standards and practices" that IBTech had to have to reach a Maturity Level 3. Whereas at Level 2, software houses could have their own codes of practices, at Level 3, all software houses would have to use the same process for a key process area—hence, the need for the GreenBook.

Thus, the 'transformation' the GreenBook involved was to have a direct impact on the execution of software development. It was no longer only something about "learning from experience" and "experimenting with new practices." The GreenBook consisted of "*Prescriptive standards* at the IBTech Equities organizational level *that everyone uses* for Project Management [and a] *Catalogues of Software Engineering practices*" [Newsletter, May 2003; emphasis in original document].

The GreenBook, in addition to being a set of prescriptive standards and catalogue of practices, was to be, "an interactive application that everybody will use to manage their projects":

> - This application will allow you to create new projects and manage existing ones by being guided through the steps of your project based on the lifecycles you have chosen. You will also be able to create new documents from standard templates.
> - Standards and adapted to different types of projects both in terms of size and software engineering methods.
> - Relevant metrics for continuous process improvement will be defined and gathered automatically through the interactive application. [Newsletter, May 2003]

In sum, what the newsletter depicted was an information system that contained the different software development lifecycles appropriate for all types of projects. Depending on the expected development time and the budgeted development cost, an appropriate lifecycle was to be imposed. The GreenBook was to be the repository

integrating all the processes needed, and containing all associated procedures, flowcharts, templates, standards, and guidelines. This information system was to ensure that all processes were completed correctly and that the project was on time and within budget.

In line with the objectives of a 'learning organization,' employees working for different software houses and regions were to take ownership of the maintenance of the GreenBook: "the processes will be kept up-to-date by allowing the owner of any new process to document it directly in the GreenBook". It was believed that the benefits arising from the 'transformation' would include:

- Individuals project teams no longer have to spend valuable time on the maintenance and conformance to their own disparate COPs.
- Provide a standard look and feel to the project management and software development process across IBTech regardless of location.
- Provide a common language that facilitates interaction between internal teams.
- New changes to IBTech practices can be rolled out globally and consistently.
- New starters can quickly be trained in "the way we do things here in Equities" and get up to speed.
- People transferring from one location to another still perform tasks in the project lifecycle in the same way.
- Assist in the achievement of CMM level 3. [Newsletter, May 2003]

The transformation was to be achieved in two phases and within an aggressive time frame:

From Q1 to Q3 2003: Phase 1 – Reengineering the GreenBook

- New contents defined and documented
- Interactive GreenBook application developed
- New ownership and maintenance model in place

From Q3 2003: Phase 2 – Leveraging the GreenBook momentum in order to become a Learning Organization

- Institutionalize process improvement: with the creation of motivated and dynamic working groups that will focus on solving problems and driving experimentation into new approaches. When the new approach is considered successful, new practices or processes will be documented in the GreenBook.
- Leverage relevant Six Sigma projects: incorporate Six Sigma project outcomes into the GreenBook

- Performance indicators: define, set-up and use metrics at the project management and software engineering levels for continuous process improvement.
- Continuous assessment of our processes and products: […] deliver CMM level 3 roadmap and compare internal service provision against industry-wide practices.
- Work with knowledge Management: to quickly and efficiently transfer the Knowledge created and capture in the GreenBook. [Newsletter, May 2003]

When the researcher joined the organization in March 2004, there was no GreenBook in place. No GreenBook had ever been near to completion. In fact, not even an embryonic version of it had been developed. Standardizing software development practices had turned out to be more difficult than expected.

5.2.3 From GreenBook Transformation to Equity Process Framework

When the newsletter presenting the GreenBook Transformation program was published in May 2003, it was clear to many corporate workers that a promise that could not be kept had been made. Developing the information system was only the tip of the iceberg; defining and documenting standard software development processes and making them part of a software lifecycle so far unknown to most corporate workers turned out to be the real challenge. In spite of the great fanfare surrounding the GreenBook, no significant results were visible.

At the end of July 2003, a solution had to be found to save face. The GreenBook Transformation was renamed the Equity Process Framework (EPF). It was hoped that the term 'EPF' would better reflect its purpose than the term 'GreenBook Transformation.' The change in term was also an opportunity to subtly narrow down the program to a more realistic scale. The July newsletters opened:

> EPF is the new name for our GreenBook, reflecting its increased scope and overall purpose. […] Good progress was made in June, completing and agreeing with management: the scope of the EPF, the blend of internal/external best practices and the approach to piloting out recommended approach. [Newsletter, July 2003]

The EPF did not conserve the idea of a grand program aimed at transforming IBTech Equities into a "Learning Organization." From that point on, the EPF was to be only one among many components of the 'learning organization' program, not the key enabler the GreenBook purported to be. Corporate workers were also much more careful in their approach, considerably less confident in their ability to institutionalize at will a standard process across the organization. Piloting processes and gaining feedbacks from process users became salient:

> Pilots have the following objectives: (1) Road-test EPF on several projects, improving EPF based on feedback during pilots. (2) Demonstrate group/personal productivity improvements. (3) Show that management and software engineering practices knit together seamlessly. (4) Show lifecycle and processes integrate well with management and engineering toolset. (5) Serve as a model for deployment of EPF into other team. [Project Brief, July 2003]

Although the EPF was less ambitious than the GreenBook Transformation program in terms of scope, it offered an additional element of complexity. The EPF was to be based on an off-the-shelf software engineering process. Whereas the GreenBook purported to formalize in-house software engineering practices, the EPF was based on Rational Unified Process (RUP):

> Equity Process Framework (EPF) will be a set of process assets for all of IBTech […] These assets being acquired to meet our business objectives and they represent investments that are expected to provide real business value. They include: (1) A set of Standard processes. (2) Description of life-cycle models. (3) Guidelines and criteria for projects to tailor the EPF's set of standard processes, balancing: (a) Flexibility to address contextual variables such as the domain, nature of the customer, cost, schedule [… and] (b) Consistency so that organizational standards […] are appropriately addressed. [Project Brief, July 2003]

The EPF represented the key element to possess in order to achieve a CMM/I level 3. Although IBTech could no longer let its software houses develop codes of practices autonomously, awareness of the balance between flexibility and consistency was recognized. It was further recognized that the standard development practices would have to allow for some variations across sites.

5.2.4 Rational Unified Process

The waterfall model is a discredited, but still popular, process for the creation of software. To follow the waterfall model, one proceeds from one phase to the next in a purely sequential manner. Thus, the waterfall model maintains that one should move to a phase only when the preceding phase has been completed and perfected. One important, yet controversial, assumption underlying the model is that the requirements can be unambiguously defined at the outset of the software project, and that those requirements will not change during the development process.

Iterative development, on the other hand, does not assume a fixed set of requirements at the inception of a software project. It allows the requirements to be refined as the project evolves. With iterative development, working subsets of the final product are reworked as the project evolves. Consequently, software professionals have to revise the requirements and the design during a project and conduct many rounds of testing. Iterative development differs from the waterfall model in that it more easily accommodates change.

RUP is an iterative software development methodology owned and distributed by IBM. The methodology is composed of four phases: inception, elaboration, construction and transition. Each phase contains one or more iterations in which several 'development disciplines' (e.g. requirements, analysis and design, testing) are revisited throughout the development process. Today, a common way for organizations to develop software iteratively is to acquire RUP and tailor the generic process to their needs. The product is sold under the form of a web site that can be deployed on an intranet.

The benefits that IBM consultants and salespersons highlighted when describing RUP captivated IBTech's corporate workers. Memos and other documentary evidence collected by the researchers show that corporate workers responded positively to the claims that RUP "provides a disciplined approach to assigning task and responsibilities," "ensures the production of high-quality software […] within a predictable schedule and budget," is a set "of industry proven best practices," and is "an enabler for reaching CMM levels 3" (IBM Rational, 2000; 2001).

Following the difficulties experienced with the 'learning organization' program, acquiring the RUP platform emerged as the way forward to quickly establish a standard software process and reach CMM/I Level 3. Consultants and salespersons demonstrated, using carefully-selected examples from the industry, that while CMM/I had become the most popular model for improving software process maturity, RUP had emerged as the de facto off-the-self software engineering process. The argument was put that RUP helped to increase the process maturity of software organizations undergoing a CMM/I initiative. According to this line of thinking, it was believed that all that IBTech would need to do would be to roll out a process platform on a server, and to progressively tailor down the generic software engineering process to reflect the organizational particularities.

The RUP platform was purchased in July 2004 to lay the foundation of the EPF. The plan was to use the generic RUP process as soon as it was deployed and to progressively customize the process using internal best practices. Thus, the identification and selection of best practices, using the 'learning organization' governance structure, was to remain an ongoing effort. It was believed that the anticipated benefits would be worth the effort:

> Example of anticipated benefits from developing and deploying EPF are:
> - It forms a common language that aids interaction between internal team.
> - New changes to IBTech practices can be rolled out globally and consistently.
> - It forms an important part of the foundation required in the achievement of CMM level 3.
> - Give a standard look and feel to the project management software engineering processes across IBTech, regardless of location of project area. [Project Brief, July 2004]

The EPF surely had the professional look and feel to give IBTech the appearance of being a mature software organization, but, as often as not, those who encountered it took a dislike to it. The generic platform was so large in scope and content that corporate workers could not distil from it anything practical for their work. Many corporate workers introduced to the EPF feared that even more documents would have to be completed because of the comprehensiveness of the RUP process.

Corporate workers involved in the decision to purchase the platform and to deploy it soon realized that the organization could not expect its members to use the generic platform. It was decided that the generic process would have to be customized first, and then would have to be launched again. This change of direction is ironic, considering that RUP was initially acquired in an attempt to avoid progressively defining a standard process.

IBM Rational consultants and salespersons had ensured IBTech that "an easy-to-use process authoring tool" would enable them to customize RUP for their precise requirements. Customizability was, in fact, one of the main selling points of the methodology:

> RUP is a flexible process platform […] With RUP Builder you can select and implement just the plug-ins that are necessary, while IBM Workbench helps you model and develop your own knowledge assets into process plug-ins […] the simple, four-step user interface helps you customize your process. (IBM Rational, 2001)

Although many individuals at IBTech were highly technologically proficient, the platform appeared rigid to those who attempted to customize it. For one thing, the process authoring tools required particular technical knowledge to be used. For another, to delete a generic process or add a new one required some knowledge of the entire RUP process. The processes were closely intertwined, and deleting a process often resulted in cancelling the creation of a seemingly insignificant artefact that was, nonetheless, refined over time into a mandatory artefact. Thus, corporate workers realized that in order to modify RUP, an extensive knowledge of the entire generic software development process was generally required.

The difficulties associated with customizing RUP and the negative reactions it evoked were the source of bitter disappointment. This state of mind was expressed in many stories heard at IBTech, some funny, others serious. Yet, in spite of all the difficulties, some corporate workers became very attached to the idea of developing software iteratively. As one of the main advocates of RUP said, "all investment banks develop iteratively and don't see why we don't. Welcome the twenty-first century."

5.2.5 Agile Methodologies

It was widely believed at IBTech that RUP was too comprehensive—a belief that corporate workers made no secret of. It was obvious that if the EPF were ever going to be used, it would have to become more accessible and less intimidating. In other words, the right level of process formality that would meet the organizational need would have to be found. In November 2004, a newsletter discussed the situation overtly:

> The Equities Process Framework (EPF) is currently based on Rational Unified Process (RUP). RUP is comprehensive and designed to be configured by an organization but there is the opinion that it could be too heavyweight for the organizational needs. There is the thought that Agile practices could be applicable in this instance. [Hot Topics, 08-11-2004]

In the business media, agile development is said to provide an antidote to bureaucratic methodologies and to re-establish software development as a creative and people-centered activity. Agile development evolved in the mid 1990's as part of the reaction against methodologies like RUP, which emphasized process definition and the production of written documentation. The agile manifesto, created in 2001, promotes four values that encapsulate the core philosophy supporting this software development approach: individuals and interactions over processes; working software over comprehensive documentation; customer collaboration over contract negotiation; and response to change over following a plan. Readers who are not already familiar with agile development can refer to Appendix 3 for an overview.

In November 2004, a SEPG Working Group, the "agile task group," was formed and given the mission to examine what agile methodologies had to offer. The SEPG Steering Committee wanted to know how agile development could be utilized within IBTech. As corporate workers became increasingly eager to better understand agile methodologies, the methodologies became part of the organizational landscape and everyday talks.

Due to the inability to customize the RUP platform and the growing popularity of agile development, corporate workers considered acquiring an agile commercial off-the-self component for the RUP platform. The component, also distributed by IBM,

would not require changing the generic RUP process or using the authoring tool. More importantly, it could allow the organization to get, overnight, a more reasonable level of process formality than with RUP.

It is important at this point to note that most corporate workers involved in the 'learning organization' program remained only superficially familiar with iterative development. While adopting a RUP platform, and, increasingly, an agile methodology, seemed to make sense, IBTech still used a waterfall process; requirement analysis, design, coding, and testing were completed sequentially. Software products were released at the end of the development lifecycle, which sometimes lasted more than a year. IBTech valued predictability and discipline, and executed software development accordingly.

Iterative development was said to enable adaptability, enable early risk mitigation, help manage complexity, and reduce defects. Agile development was said to foster innovation. What IBTech sought, however, was to institutionalize a standard software process across several sites. In other words, it was the idea of an out-of-the-box standard that had made RUP and agile methodologies appealing, not their technical merits.

IBTech relied heavily on consultants to comprehend and implement practices. In an attempt to better assess how agile practices could be applied, corporate workers' immediate reaction was to hire the services of Bill Jobs (fictional name), a well-known agile consultant and author of the agile manifesto. At IBTech, Jobs was solemnly referred to as 'the guru.' In a tone which echoed that used previously by the IBM consultants, Jobs delivered a series of lectures at some of IBTech's sites, including Asia, America, and EMEA.

The key idea underlying Jobs' presentation was that software development is new product development. The consultant defended the argument that software development is an activity that requires adaptive steps driven by 'build-feedback cycles.' Consequently, in Jobs' view, formulating a plan and estimates for a whole project at its inception was nonsense. Using the counter analogy of mass manufacturing, the consultant argued that "a waterfall lifecycle, big up-front

specifications, estimates, and speculative plans applicable to predictable manufacturing have been misapplied to software projects, a domain of inventive, high-change, high-novelty work."

Being aware that his discourse might not appeal to a management audience, and, indeed, might be perceived as anti-managerial, Jobs adroitly managed to give agile development a distinctive strategic flavor. An electronic presentation slide read as follow:

> Agility […] is about succeeding and about winning: about succeeding in emerging competitive arenas, and about winning profit, market share, and customers in the very center of the competitive storms many companies now fear.

The consultant also fervently attacked the CMM/I and other means of formalizing the software process. Quoting a study on the productivity gains offered by the CMM (Harter et al., 2000), Jobs presented a slide, written in bold, that said that "Increasing CMM process maturity was associated with lowered productivity."

A corporate worker actively involved in the 'learning organization' program interrupted the speaker during his attack on the CMM/I and plainly asked him to emphasize the benefits of agile methodologies, rather than the problems associated with formalism. The content of the presentation had become an embarrassment for her and several of her like-minded colleagues in the audience.

Jobs' presentation generated much talk at IBTech. A consensus regarding the importance of institutionalizing a standard software process existed within the organization. However, every time an effort was made in this direction, other equally important beliefs were shaken. The CMM/I lead assessor felt betrayed by the argument publicly presented by Jobs, as did many CMM/I advocates. The content of the presentation was a blow for many.

5.3 Software process improvement

Software process improvement is a pervasive idea within software engineering. It seeks to improve the execution of the software development process – reducing cost associated with discovering and fixing software defect, improving the productivity of developers, eliminating unwanted deviations from standard processes, etc, and, ultimately, improving product quality. In the organization studied, it was believed that a software process improvement program "delivered little value without a systematic approach and technique for correcting processes" and that learning could only occur when processes "are identified, measured, and controlled" [A43].

5.3.1 The post-implementation review

The post-implementation review (PIR) provides an ideal context in which to examine software process improvement in relation to the standardization of software development practices. This is because the PIR was both a standard software process and a key component of the 'learning organization' program (see Table 6). A presentation of the PIR is, at this point, useful for establishing details regarding the process by which standard software practices should, in theory, have been created, and may illustrate the difficulties associated with institutionalizing standard processes across IBTech. The SEPG Steering Committee wanted to make the PIR the first standard software process institutionalized across the ten software houses. However, the PIR was never fully institutionalized and, interestingly enough, was further de-institutionalized while the researcher was carrying out fieldwork.

The PIR corresponds to what is also sometimes called a 'project retrospective' or a 'postmortem review.' Since IBTech did not develop software iteratively at the time the research was carried out, the completion of the PIR marked the end of a project. The following extract from the process guidelines highlights the objectives of the PIR:

> The PIR objectives are to identify and share areas for improvement and lessons learnt; […] to highlighting success stories and best practices used during the project; to better capture and communicate the business and technology satisfaction level on the project; to prioritize the process

> improvement actions on a continuous basis through the Software Process Group (SEPG). [PIR process definition, 03-08-2004]

The project manager was responsible for all steps of the PIR. The process involved creating a web-based survey questionnaire and sending it out to the project participants. The survey was the instrument used to collect information about completed projects without compromising the confidentiality of the project participants. Respondents included the project sponsors, the users of the system, the project managers, the environment team, the release team, and software developers. Depending on the types of project, the project manager might have to send two different survey questionnaires for a single project, one covering principally technical questions, the other covering business questions.

The project manager was responsible for compiling the survey results and producing a provisionary PIR report. At that stage, the project manager also had to track down and integrate into the provisionary report three categories of project metrics (costs metrics, schedule metrics, scope metrics). The survey results and the metrics were the information to be discussed in a PIR meeting. The objective of the meeting was to allow the project participants the opportunity to give feedback about the project. The project manager had to take note of the important points raised during the meeting. Following the meeting, the project manager had to produce the final PIR report, which presented clear areas for improvement, lessons leaned, and best practices. A PIR summary report also had to be produced by the project managers for high-profile projects. The final step of the PIR was to transmit to SEPG the areas for improvement, lessons leaned, and best practices via a workflow management system called the 'SEPG CD system.' The PIR report also had to be made available to all employees on an intranet and, in some cases, presented to high-ranking executives.

Only a bird's eye view of the PIR process is given here. The flowchart describing the process and the documents to be produced at each step of the process could cover a whole wall. The PIR was of such complexity that milestones had to be established for it in a process management system. The distribution of the surveys and the transformation of the survey data into information were, by themselves, projects of their own. It was quite common for project participants to ignore emails requesting

them to complete the web-based surveys. The response rate was sometimes so poor that project managers had to call respondents in person to encourage them to fill the surveys out. Project managers who had two types of questionnaires to deal with for a single project assumed a heavier bureaucratic burden.

The PIR process was standard across IBTech when the researcher entered the organization, but was not fully institutionalized. The PIR was vehemently criticized for being too rigid and bureaucratic. The following statement illustrates the project managers' general feeling vis-à-vis the process:

> The Post Implementation Review is fundamentally flawed. It is not possible to create milestones in advance for when a post implementation review should take place as it is not possible to state when the application/product will be finally implemented. Also, the process is over-engineered. [Project manager, Call raised to SEPG 07-08-2004]

Even though the PIR was recognized as being a key component of the 'learning organization' program, some project managers energetically attempted to by-pass the PIR. One common strategy was to ignore the PIR and claim to have other priorities. The following thread of emails shows how this was done:

> Hi Gerald [Project manager],
> Following on from our previous conversation – please can you let me have an update of your PIR […] [Project manager officer, email 19-07-2004]
>
> have done nothing at all… too busy
> are you sure I have to do one? [Project manager, email 19-07-2004]
>
> Gerald – we have been over this and it is a definite requirement. [Project manager officer, email 20-07-2004]
>
> What am I meant to do with this reply??!!?? [Project manager officer to manager responsible for the PIR, email 20-07-2004]
>
> Gerald,
> Yes, a PIR must be completed for all demands and initiatives. [Manager responsible for the PIR, email 21-07-2004]
>
> fine – well haven't done it. [end of the email] [Project manager, email 23-07-2004]

Another strategy used by project managers to get around the PIR was to periodically report the creation and distribution of the internet-based survey in the workflow management systems (the 'SEPG CD system') used to monitor the progress of a PIR. In one case, the project manager reported the target date of the PIR of one month eleven times over a year. A year after the completion of the project, the PIR was finally cancelled as none of the project participants could remember enough about the project to comment on it.

The corporate worker responsible for the PIR routinely faced hostile reactions in relation to the process. However, instead of making the PIR less exhaustive or allowing the project managers to identify areas for improvement, lessons learned, and best practices in a way they found appropriate (i.e., de-standardize the PIR), the corporate workers preferred to play with its presentation in the hope that it would appear less bureaucratic:

> I have even reduced the number of pages in our PIR report template to make it less intimidating. Many still see the PIR, and in particular PIR documentation, as an unnecessary overhead, even though there is little effort involved… it's a culture change we are pushing at the moment. [Manager responsible for the PIR, email 06-08-2004]

It is ironic that the process meant to enable process improvement did not lend itself to improvement. According to documentary evidence collected, process improvement was, by and large, understood in the study organization as the enhanced standardization of software development practices. In accordance to the logic of the CMM/I, process improvement is about process definition, process documentation, and process optimization. Seen this way, having stable processes is the key to predicting the outcomes of projects, which is the key indicator of maturity, according to the CMM/I. So, when the corporate worker talked of "pushing a cultural change" in the earlier quotation, he meant having corporate workers accept that software development is a formal, disciplined activity.

Corporate workers at IBTech generally agreed that software development should be disciplined and predictable. Corporate workers liked to know what was expected of them in terms of deliverables and schedule, and they liked to meet these

expectations. However, they were not always ready to devote their time and energy to accepting the practices that the logic of the CMM/I presupposes. Could the PIR have pushed the level of bureaucracy that corporate workers were able to tolerate too far? Could corporate workers not have seen the connection between the PIR, learning, and the establishment of more disciplined and predictable software development practices?

In August 2004, the CMM/I lead assessor surrendered to the increasingly-frequent and vehement attacks delivered against the PIR. He had been convinced that the PIR was unnecessarily bureaucratic and standardized, and, in an email addressed to Chief Business Technologists of the three regions (Asia, EMEA, North America), called for a bold process change:

> Software Houses are free to develop and implement their own PIR process. […] The PIR Report must include good working practices, lessons learnt, and recommended changes to existing processes. […] Improvement to local Software House practices will be identified and documented in the Software House and/or Code of Practice. [CMM/I lead assessor, email 01-08-2004]

The implications of the process change were two-fold. First, the change meant that software houses would no longer have to use the standard PIR process. Software houses could use a survey other than that sanctioned by IBTech—one which was possibly much shorter-- or no survey at all. For example, a project manager officer based in Asia suggested that an informal meeting should be sufficient to identify good working practices and develop the global software processes:

> The PIR report isn't something that people in Asia really look at or are managed against. For that reason, it is ineffective right now. We want to ensure that our pms [project managers] are incentivised/held accountable for delivering pirs but so far we can't justify tasks which don't deliver the value they ought to. […] My view is that a regular […] review is the right forum in which to discuss [process improvement]. [Project manager officer, Email 07-08-2004]

More significantly, however, the CMM/I lead assessor suggested that software houses should keep building their own code of practice-- that is, managing their own,

local software development methodology. In this sense, the email can be seen as a statement in favour of a plurality of understanding of what software development is, and how it should be practiced. Thus, it is a move in favour of de-standardization.

5.3.2 The Software Engineering Process Group

As was previously explained, the 'learning organization' governance structure provided a framework for ensuring that novel practices and process improvement opportunities discovered in a particular software house become known and applicable in other software houses. SEPG played a key role in this framework; it created a synergy among software houses, allocated resources to pilot and implement new processes, established process standards, and, more generally, promoted a learning mindset in the organization:

> The Software Engineering Process Group (SEPG) is the forum to facilitate the definition, maintenance, improvement and establishment of the software engineering and management process for IBTech, allowing Areas for Improvement and Best Practices to be evaluated, and developing a culture of Continuous Process Improvement. SEPG has a direct correlation with two IBTech Equities vision principles: Provide Excellent Solutions With Greatest Efficiency and Be a Learning Organization and Innovate. [SEPG description document, 06-14-2004]

A PIR process was considered completed only when the process improvement suggestions captured in the final PIR report were transmitted by a project manager to a regional SEPG Definition Group. Process improvement suggestions were transmitted to the SEPG Definition Groups through a workflow management system, the 'SEPG CD System.' A member of a SEPG Definition Group then ascribed a priority level to the improvement suggestions based on their perceived complexity and relevance. 'Big tickets' were used for improvement suggestions that were deemed important, and that were likely to require a considerable commitment of time to be institutionalized. 'Small tickets,' on the other hand, were used for minor changes to an existing process.

Following the 'learning organization' governance structure presented in Figure 2, big tickets were given priority and submitted to the SEPG Steering Committee for

further review. The SEPG Steering Committee could choose to pilot a new practice or process within a set time and budget constraints. Depending on the pilot outcomes, the SEPG Steering Committee could establish an implementation plan for the new process to be institutionalized. Standard software development practices, according to organization-sanctioned documents, were institutionalized at an organizational level when incorporated into the EFP. Small tickets were, as a general rule, handled by the regional SEPG Definition Groups and did not require the involvement of the SEPG Steering Committee.

The SEPG was often criticized for being highly bureaucratic. Although the project manager who submitted a process improvement suggestion on the behalf of project participants should, in theory, have expected a response from the SEPG within two weeks, the response was usually only an acknowledgement that the improvement idea had been received. Once a process improvement idea was entered into the SEPG CD System, it disappeared into a bureaucratic black hole. Despite the effort needed to complete a PIR and identify process improvement suggestions, it was not rare that an improvement idea would stay in the Equities SEPG CD System for more than six months without being proactively dealt with [A26, 28-06-2004; A28, 01-07-2004].

Members of the SEPG Definition Groups, who were responsible for transmitting critical improvement ideas to the SEPG Steering Committee, themselves showed little interest in the bureaucratic SEPG. Until July 2004, meetings were cancelled week after week. At some point, the corporate workers involved in SEPG did not even bother to cancel, leaving the Chair of the weekly meeting alone. An email with the heading, "3rd Consecutive SEPG Definition Group Meeting – CANCELLED," opened,

> We have a weekly SEPG at 9 AM (EST) every Wednesday. 2 weeks ago this meeting was cancelled on the day of the meeting because a number of people were attending a conference. As people probably planned to attend the conference, we could have planned in advance to reschedule this meeting for another date. Last week, this meeting was cancelled on the day of the meeting with no stated reason. This week, I did not receive a meeting cancellation notice or any other notice, so I again assumed that the meeting was on. However, none dialed in to the conference, so it appears to be cancelled for the 3rd week in a row. [Quality Leader – North America, Email 30-06-2004]

In July 2004, a member of the SEPG Steering Committee made use of her authority to reinvigorate the SEPG meetings, but later acknowledged during a one-to-one meeting with the researcher that the SEPG had been proven unviable. She observed that although most corporate workers agreed that learning was important and fostered innovation, few seemed to agree that it should take place according to the 'learning organization' governance structure. In her view, all that SEPG did was to perpetuate the myth that learning had to occur through a formal and systematic process. It was becoming clear to her that corporate workers had a different view of how software development should be practiced, but she could not pin down what it was.

This chapter has provided an overview of the evolution of the 'learning organization' program and of the context in which software development took place. The chapter which follows will present the organizational discourses that circulated at IBTech, and the role they played in the constitution of 'software development.' This chapter will also take a closer look at some particularities of the context that are, analytically, highly relevant.

6 Empirical Findings

This chapter outlines the data collected in the study organization over a ten-month period, from March 2003 to January 2004. The data were mainly obtained from participant observations and organizational texts. Participant observation data are systematized into three themes characterizing the organizational context: hierarchy and status, bureaucracy, and consumption practices.

More than six hundred relevant organizational texts were analyzed in order to identify the discourses that are at work in them. The analysis has revealed that seven discourses form two distinct representations of the object 'software development.' The hermeneutic of suspicion was applied in both the analysis of field notes and organizational texts.

6.1 Participant observations

An in-depth immersion in the organization was necessary to understand the context, with 'context' referring to the wider environment in which corporate workers are situated. Thus, by spending considerable time (40 to 50 hours per week over 10 months) in the organization, the researcher not only acquired a sense of what it is like to work at IBTech, but also refined his knowledge of what it is like to work for an investment bank and in the corporate world. Acquiring an understanding of the context was necessary to interpreting the organizational texts and comprehending the discursive practices of corporate workers. As Bourdieu notes, by focusing exclusively on the texts produced, transmitted, and interpreted by individuals in a particular setting, as some discourse analysts sometimes do, an analyst is bound to miss the question of why individuals say what they say, in the way in which they say it (Bourdieu, 1991: 44 & 237; 1992: 149).

The following insider account presents three distinctive characteristics of the organizational context. These characteristics will help at a later stage to explain how software development is constituted at IBTech and why it is constituted the way it is. These characteristics include the existence of an ongoing struggle for status elevation

and conservation, the prevalence of a bureaucratic ethos, and the role played by consumption as a means of communicating the possession of a valued set of norms.

6.1.1 Status elevation and conservation

IBTech has a traditional hierarchical structure. The distance between hierarchical levels informs how interaction takes place between individuals. If more than one hierarchical level separates individuals, interaction will, as a general rule, take place through the formal communication channels provided by the hierarchy and involve intermediary managers. However, in parallel to the role and authority structure provided by the hierarchy, there exists in the organization an informal status structure. Status, as it is understood here, is highly consensual and determined by the recognition one receives from others.

The recognition one receives from others is determined by the extent to which one's behavior is perceived to be appropriate. More specifically, one gains recognition from others by acting consistently over time in accordance to the normative standard of behavior that prevails in the organization. In IBTech's language, the term 'professional' is commonly used to encapsulate this general idea of 'appropriateness.' Possessing a high status results, according to a corporate worker, in one being "more popular with managers and more influential with colleagues."

It is important to note that the existence of an informal status structure does not go against the authority structure and the mode of communication established by the formal hierarchy. In fact, the informal status structure contributes in the present case to reinforcing the formal hierarchy: it leads individuals to act in accordance to the formal procedures in order gain recognition.

Corporate workers show remarkable status consciousness. Indeed, an important aspect of their jobs appears to involve cultivating a professional identity and seeking to obtain the recognition of superiors and colleagues. The desire to improve one's status is deeply seated in the organizational life and is in evidence in several practices. In order to elevate their status, corporate workers must demonstrate that they possess a relative degree a mastery of the organization's de facto way of doing

things. This entails, for example, addressing other corporate workers with an adequate level of formality and impersonality, showing interest in the work itself, and, more generally, showing commitment to the organization. Failure to conform to the prevailing way of doing things typically leads to a person and his work being perceived of as being unprofessional or plain careless.

It is important to note that the pursuit of status is so deeply-seated in organizational life that it does not seem to operate at a conscious level. Corporate workers do not seem to be consciously strategizing their next move so as to elevate themselves on the organization's honour scale. In a related vein, what the corporation expects from its employees is not fully explicit. As one corporate worker told the researcher, "There are a million little rules to obey." Corporate workers, nonetheless, understand with varying degrees of sophistication what is expected from them and what practices can provide them with the recognition of others.

So, corporate workers do things because they are the right things to do. To put the matter differently, they often perform a particular action simply because they sense it is the right thing to do for the organization and themselves. The following dialogue between the researcher and a corporate worker that took place during an interview is illustrative of this fact:

> *The researcher*: How do managers know you've got what it takes [to succeed professionally]?
>
> *Corporate worker*: Because I show that I really want to.
>
> *The researcher*: How do managers know that you work well?
>
> *Corporate worker*: Because I'm among those on this floor that put in the longest hours and spent the most time at work.

The logic to which the individual subscribes is, of course, that the quality of the results is proportional to the work put in. However, in this case, it is clear that it is not only achieving high quality results that is the source of motivation. The answer given to the researcher's questions suggests that the individual is also concerned with communicating that he is willing to place the interest of the corporation before his own interests. The corporation expects commitment from its employees, and working

long hours and adopting the normatively-sanctioned way of conducting oneself are prime signs of devotion.

Obvious acts of professionalism are also apparent in several other organizational texts. Every fortnight, IBTech releases a memo entitled, "A day in the life of…," intended to present a typical workday of an employee. The texts, which are always written by the employee profiled, are interesting in that they illustrate the perception corporate workers would like others to have of them. Moreover, these texts richly capture the correspondence between what the corporate workers feel is expected of them and the extent to which they are willing to meet these expectations. Reference is usually made to the long period of time spent at the office and to attendance to meetings where highly ranked corporate workers (e.g. the CTO) are present. Here is a typical example:

> My alarm clock starts to go off at 5:15 AM. […] I usually get to work around 6 AM […] When I get to my desk, I check my email for any Production issues, which require my immediate attention. Then I look for any issues Asia or London connectivity teams might have escalated to us (if they didn't already wake me up in the middle of the night) […]
>
> 8:30 AM, Market Open. The hotline may start ring a lot more now. OK, so I'm done with the morning issues […] for the next 6+ hours I'm on a billion conference calls […] In between these meeting, we are monitoring our FIX Connections for their connection status […] Throughout the day, I *may* find time to grab some food to bring back to my desk and eat.
>
> 3:00 PM Market Close. End of day issues start rolling in. […] I could be out of door anywhere between 7 PM –12 AM. [Newsletter, 09.12.2004]

The inculcation of professional norms and an interest in status elevation and conservation is not purely incidental. An individual working in the area of human resources mentioned to the researcher during an informal conversation that her role involved developing the mechanisms required to make corporate workers go beyond the minimum of their contractual relationship with IBTech. She lamented that it was difficult to convince technical contractors to adopt the desired patterns of behavior because they did not feel the pressure to conform as much as permanent employees. In her view, the problem was mainly caused by the fact that technical contractors,

who are often with the organization for set period of time, did not see the point of seeking to elevate themselves in the organization's honour scale.

A discussion of why conformity matters so much from the point of view of the organization cannot be attempted here. The important point to note for now is that corporate workers are inclined to conform to the norms of professional behavior that prevail because conformance leads to status elevation. The enactment of the corporate norms in routine work practices determines in large measure the status of corporate workers. In this sense, IBTech's corporate workers answer to an ideal that is not so different from the ideal of the chivalry epitomized by the pursuit of glory and repute.

6.1.2 Bureaucracy

IBTech is remarkably bureaucratic. Information that is not documented is typically regarded as anecdotal and, hence, inadequate for supporting action. This attitude towards written documents may be explained in terms of the ideal of professionalism that corporate workers share. Corporate workers value that which appears formal, exact, and rational, because it appeals to their sense of professionalism. The mere act of documenting information has the power to confer it with greater veracity and the document producer with appropriateness.

The high level of bureaucracy in software development is at IBTech justified by the fact is that investment banking involves a significant sum of money. One organizational member put the matter this way: "The reason we celebrate success with such enthusiasm is because we know how costly a mistake can be. […] As a bank, we need to have policies and procedure because there's risk embedded in almost everything we do." Corporate workers involved in software development prefer to think of themselves as investment bankers rather than as IT professionals. An informant once remarked in a conversation that "You have to look like an investment banker." In the corporate worker's mind, investment bankers are disciplined and formal in their approach to work. Adhering to policies and procedures, and producing documents with care, contributes to reinforcing the sense

of professionalism that is expected of an investment banker, i.e. someone dealing with phenomenal sums of money.

Producing written documents in accordance to the policies and procedures is, on the whole, seen as professional, while seeking to gain autonomy over one's work is unprofessional. Because status is at IBTech given great importance, and because complying with the bureaucratic way of doing things is associated with the behavior of those of a high status and an ideal of professionalism, organization members tend to comply with the bureaucratic procedures. There are, however, exceptions. In the following excerpt, a corporate worker contests the idea that for a best practice to be learned, it has to be identified by means of a PIR and documented:

> 'Lessons learned' aren't learned, they are just written down. They are in the head of the best people. PIR is a joke; there is just no time for that. A team must quickly attack a new project; there is no time for discussing metrics. Senior people possess a massive amount of knowledge. [Project manager, 11-05-2004]

Bureaucratic annoyances at IBTech are balanced by the obvious pride many people take in working for a leading investment bank. These annoyances are seen as being part of the experience, as observed by one individual: "It's necessary evil, and if you enjoy your job you've got to enjoy everything that comes with it." So, bureaucracy should be understood as an important aspect of the life at IBTech, and acceptance of the bureaucracy as an acceptance of this life.

One thing that corporate workers do not tolerate is being requested to complete documents by someone who does not occupy a position superior to theirs in a direct hierarchical relationship. This request is likely to be perceived as an illegitimate subordination attempt, and even an outrage to the norms of the corporation. As explained above, the authority structure provided by the hierarchy determines the span of control that a corporate worker has over the work of others and the nature of their professional relationships with other corporate workers (i.e. either subordinate or superior). This mode of communication, which takes place principally between superiors and subordinates, complicates the task of those involved in standardizing

software development. Quality representatives very seldom have the authority to request that others take part in software process improvement.

6.1.3 Consumption practices

Employees of investment banks are usually better paid than individuals working in other sector of activities, especially those to a lesser degree connected to the corporate world. In the United Kingdom, graduates with a Bachelors or Masters degree begin as Analysts on around £35,000 base salary. Considering that the average starting salary for all graduates and for graduates in the top 100 AGR blue-chip firms are £15,500 and £23,000, respectively (BBC, 2006), a salary in the mid-thirties is relatively high. Those with MBAs and some prior work experience can expect to begin as Associates at around £60,000.

Folk rumour has it that employees of investment banks receive an important portion of their total compensation in bonuses. This is only sometimes true for employees of 'front office' divisions, and almost never true for employees of 'back office' divisions such as IBTech. In spite of the virtual absence of bonuses in IBTech, corporate workers earn enough to engage in non-utilitarian consumption. For example, a corporate worker confessed to the researcher during an informal exchange that the act of consumption in any form was her way "to take some stress out." The shopping bags she was carrying at the time the conservation took place suggested that she had had a successful shopping excursion to a well-known fashion retailer targeting female professionals.

At IBTech, several practices involving the consumption of products and services have the power to communicate an understanding and acceptance of the prevailing norms. As cliché as it might sound, choice of dress, hairstyle, reading, entertainment, etc. have the power to signal acceptance of the prevailing norms. In most cases, the products and services corporate workers consume appear to reinforce their professional identity rather than individuality. Considering that the status of individuals is determined in large part by the ability to communicate to others their acceptance of the norms of the corporation, it becomes understandable why consumption practices are so significant.

It is important to note that fieldwork data do not suggest that the status of corporate workers is determined by their pecuniary strength. In the present context, consumption is instead understood as an act resulting from, and being used to communicate, the possession of a particular set of norms. Thus, the amount spent is not as important as the accepted symbolic value of goods with which one chooses to surround oneself. It is also important to note that fieldwork data do not suggest that corporate workers consume goods in order to (i.e. with the conscious intent to) communicate that they are committed to their organization. Rather, corporate workers appear to possess an acute sense of the kind of actions that can position in the informal organizational order.

The attitude towards consumption that corporate workers display in their private lives is also reflected in their professional functions. This fact has critical implications for the present study. Corporate workers make their organization acquire several management techniques over a relatively short period of time, in some cases without really knowing what they entail in terms of practices. Observations suggest that the ideal of professionalism and rationality that consultants and vendors offer is often rich in meaning and very seductive to corporate workers.

In one particular case, it took only a few minutes for an individual to determine that a particular management technique, until then only remotely known to him, was required. The CD demo received from a vendor had been sufficient to determine that Rational Unified Process ideationally converged with what the organization valued. The demo presented impeccable interfaces that signalled rigor and discipline. The individual's approval of the product presented him with the means to show allegiance to the norms of the corporation – the very norms that he was expected to adopt and maintain to maintain his status.

For corporate workers, questioning too forcefully the appropriateness of management techniques which promise the sort of things the organization values can have negative consequences. It may be perceived as an act of resistance to the norms of the corporation and, equally harmful professionally, as an inability to adopt the prevailing norms. During sales presentations, nodding to signal agreement to an ideal of professionalism found in the thickness of the reports, and to the veneer of order

and discipline found in commercial off-the-shelf software process, is often to best thing to do from a career point of view.

6.2 Organizational texts

More than six hundred organizational texts were collected and analyzed for this doctoral research. The term 'text' is used here to refer to assemblages of oral and written forms (Putnam and Cooren, 2004). Thus, mundane conversations among co-workers are considered to be as analytically relevant as formal organizational documents. The objective of the analysis was to systematically identify the organizational discourses that inhabited the texts that corporate workers produced, transmitted, and consumed as part of their routine daily activities and, ultimately, cast light on how software development is constituted through the discursive practices of corporate workers.

Organizational texts analyzed were classified by themes. Proceeding inductively (Miles and Huberman, 1994), the researcher did not begin the analysis of data with a set of themes in mind, but rather let these themes emerge. Themes were progressively delineated as organizational texts were iteratively re-examined. It was not uncommon that organizational texts superimposed, knotted into one another, and were found to belong to more than one theme. In keeping with the discourse analytic approach adopted, these themes were considered to be organizational discourses in their own right; that is to say, sets of statements that bring organizationally related objects into being (Grant and Hardy, 2003: 6). A summary of the seven organization discourses identified is presented in Table 8.

This section presents snippets of organizational texts in order to introduce the seven organizational discourses discovered. This section also highlights the tensions and connections that exist among discourses and the contradictions that exist within discourses. Significantly, an analysis of the manner in which seven discourses relate to one another has revealed that discourses constitute two conflicting objects 'software development.'

Table 8: Seven organizational discourses

Organizational discourses	Explanation
DISCIPLINE	The 'discipline' discourse stresses the importance of adopting an approach to software development in which the roles, the responsibilities, and the documents to be produced at each stage of the process are clearly specified. The discourse implies commitment to such an approach and rigor.
PREDICTABILITY	The 'predictability' discourse generally refers to the ability to accurately estimate the costs and schedule for a project before it has begun. The discourse also occasionally refers to mediating risks throughout projects.
MEASUREMENT	The 'measurement' discourse emphasizes the importance of specifying metrics and collecting information about processes in order to find ways to improve them.
BUREAUCRACY	The 'bureaucracy' discourse is about the elimination of activities that slow down the software development process and that contribute little to end results. The discourse implies efficiency.
COMMUNICATION	The 'communication' discourse stipulates that sound communication among parties involved in software development reduces development time and effort, and fosters the transfer of knowledge. This discourse principally centers on person-to-person communication, rather than on written communication.
INNOVATION	The 'innovation' discourse typically depicts software development as an activity characterized by rapid change and emergence.
LEARNING	The 'learning' discourse generally emphasizes the idea that software development is dependent on the ability of individuals and organizations to acquire knowledge. The discourse also occasionally refers to the ability to systematically improve a software process by means of software process improvement.

6.2.1 Discipline

The 'discipline' discourse surfaces in organizational texts on a broad range of topics. The following excerpt from an internal report shows that the 'discipline' discourse inhabits organizational texts on RUP:

> RUP is a proven methodology used by industry leaders. It provides a disciplined approach to assigning tasks and responsibilities. The plan is to utilize RUP to help IBTech keep track of the evolution of ours projects… [Internal report 29-08-2004, A67]

A 'disciplined approach' is defined in the same text as a software development approach in which the 'who,' 'what,' 'when,' and 'where' of software development are explicit and followed to the letter. RUP allegedly instills discipline into software development by answering the following questions:

> - What are the processes in terms of activities, roles, and artifacts?
> - Who are associated with the key processes?
> - When should the processes be initiated?
> - Where can artifacts and templates be found?
>
> [Memo 29-08-2004, A67]

It is in organizational texts appertaining to the CMM/I that the 'discipline' discourse is, perhaps, most manifest. The concept of institutionalization, by referring to the commitment and consistency to performing a standard process, clearly implies 'discipline.' Here is an extract from the CMM/I reference book used by IBTech corporate workers:

> "Institutionalization" is an important concept in process improvement. When mentioned in the generic goal and generic practice descriptions, institutionalization implies that the process is ingrained in the way the work is performed and there is commitment and consistency to performing the process. (Chrissis et al., 2003: 33)

Similarly, a consultant report obtained by IBTech from Gartner, a technology-related research firm, emphasizes the model's potential to generate discipline:

> At its core, the CMM is a model of organizational development and change… During the evolution through the five maturity levels, development practices are transformed from an ad hoc, undisciplined state into disciplined processes capable of predictable results. [Consultant report 15-02-2004, A124].

6.2.2 Predictability

The 'discipline' discourse is closely related to the 'predictability' discourse. The more disciplined a process is, the more predictable the outcomes should, in theory, be. 'Predictability' thus implies an ability to carry out application development and maintenance work on schedule and on budget.

The primary objective of the CMM/I is to achieve optimal repeatable processes for software development. Maturity level 5 focuses on continually improving performance, a goal made possible by having processes that produce predictable results. Maturity, therefore, implies predictability:

> Since improved organizational maturity is associated with improvement in a range of expected results that can be achieved by an organization, it is one means of predicting the general outcomes of the organization's next project (Chrissis et al., 2003: 81-82).

The 'predictability' discourse is implicitly brought up in the description of a 'managed process' (Level 2). A 'managed process' is a process that is executed in accordance with the plan and that results in the achievement of a specific objective established for the process, such as cost, schedule, and quality objectives. The text segment 'consistent performance' here implies, to some degree, predictability: "A managed process achieves the objectives of the plan and is institutionalized for consistent performance" (Chrissis et al., 2003: 34).

'Predictability' is explicitly addressed from Maturity Level 3. The benefits reaped from the commitment to and consistency of performing the process in a qualitatively predictable manner pave the way for subsequent maturity levels:

> A critical distinction between maturity levels 3 and 4 is the predictability of process performance. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques,

> and quantitatively predictable. At maturity level 3, processes are typically only qualitatively predictable (Chrissis et al., 2003: 81).

The 'predictability' discourse also inhabits organizational texts about IBTech's standard software development process. The following excerpt from an electronic presentation file used by members of the quality teams shows that RUP is linked to the 'predictability' discourse:

> Rational Unified Software:
>
> - Provides guidelines for efficient development of quality software.
> - Reduces risks and improve predictability.
> - Promote common vision and culture. [...]
>
> This standard software process ensures predictability for IBTech in meeting schedule, cost and high quality standards. [Electronic presentation file 10-08-2003, A123]

However, it appears that the notion of predictability is understood in the organization in a way that differs from what RUP suggests. The term 'predictability,' when referring to an iterative methodology, generally refers to mediating risks and avoiding the drama and unpredictability of 'big bang' release by delivering multiple releases of a product:

> On every project, you want to minimize risks, ensure predictable results […] Using RUP's iterative approach to development, project managers can more accurately gauge progress by assessing key milestones at each iteration – increasing the predictability of the entire development effort. [IBM-Rational white paper, 2001]

The foregoing quotation from a white paper found at IBTech explains that with an iterative methodology, there is a constant reworking of the plan with each iteration. Thus, the closer the end of project, the more the organization should know about what the outcome will be. Seen this way, 'predictability' does not refer to the ability to accurately estimate costs and schedule for an entire project before the project has begun. A corporate worker particularly knowledgeable about agile and iterative development explained the different meanings the term 'predictable' may take:

> Predictability in terms of plan-driven means that if you draw up a plan which says 'we will analyze, develop, implement' and that says 'we will deliver on August 6th' we know what we are doing and will know if we are off course. That's one sense of 'predictability' where you are trying to map out the future. Given that you can't have any control over the past, you are trying to exert control over the future. And that's a particular engineering perspective on the world.
>
> Agile and iterative development take it at a totally different level, which say 'try not too much about controlling the big future, think about how you might control the next week, the next month and work at controlling in small slices'. That way you will achieve the level of control over the future that will mean that you don't get into messes where after eighteen months work you discover that you will not have any chance of making the delivery. [Interview, 14-12-2004]

Corporate workers were uncomfortable with the idea of developing iteratively, as it suggested that cost and schedule could not accurately be predicted. The following except from an email shows this discomfort:

> [The CTO] agrees that RUP can help us reduce risks, but I'm not sure how he sees his BTs [Business Technologists] kick-off a project without knowing upfront how long the project is going to last […] it's [iterative development] a double-edged sword […] [Email from the head of governance, 23-11-2004]

6.2.3 Measurement

The 'measurement' discourse was principally found in documents highlighting the importance of specifying and collecting standard metrics across IBTech. A Metric Working Group was in place to fulfil this purpose:

> The problem of inconsistent and fragmented metrics for projects, programmes, and vendors affects all of us. Since each team collects metrics in a different format there isn't an easy way to compare metrics across teams and regions. Due to all of the above it is difficult to measure achievement of organizational goals.

> The objective of the Metrics Working Group is to define a consistent set of metrics that will be collected and reported by AD [application development] team […] Each metric will relate to one or more organizational goals that have been defined by the CTO. [A41]

A frequently-repeated statement at IBTech is that metrics should be developed to support management information needs and, ultimately, enable process improvement. This vision of software development concurs with that of the SEI:

> The primary reasons for doing measurement and analysis is to address identified needs and objectives. Measurement results based on objective evidence can help to monitor performance, fulfill contractual obligations, make informed management and technical decisions, and enable corrective actions to be taken. (Chrissis et al., 2003: 256)

The SEI also clearly specifies what measurement and analysis involve:

> The Measurement and Analysis process involves the following:
>
> - Specifying the objectives of measurement and analysis such that they are aligned with identified information needs and objectives
> - Specifying the measures, data collection and storage mechanisms, analysis techniques, and reporting and feedback mechanism
> - Implementing the collection, storage, analysis, and reporting of the data
> - Providing objective results that can be used in making informed decisions, and taking appropriate corrective actions (Chrissis et al., 2003: 247)

The importance granted to the CMM/I objective contributed to making the 'measurement' discourse circulate at IBTech. In many cases, organizational documents about measurement were inspired by the CMM/I reference books quoted above. For example, here is an extract from an organizational text written by a corporate worker:

> The goal of the Metrics Working Group is to develop a measurement capability that is used to support the Management information needs.
>
> [Internal report 09-01-2004, A41]

And here is an extract from the text that inspired the author of the text above:

> The purpose of Measurement and Analysis (MA) is to develop and sustain a measurement capability that is used to support management information needs (Chrissis et al., 2003: 247)

The SEI provides a wealth of resources to talk about 'measurement.' In an electronic presentation document, presumably to add weight to the claim that measurement matters, a corporate worker used the results of a survey completed by the SEI:

> A further survey described the potential gain of 35% in productivity, 19% reduction in defects, and 19% reduction in time-to-market […] High maturity organizations have demonstrated that the key to process improvement success is to be able to measure the process in order to make informed decision making. [Electronic presentation file 02-03-2004]

The PIR process described in the previous chapter is intended to measure variations to the plan, in terms of days and costs, and to identify the causes of such variations:

> The PIR delivers little value without a systematic approach and technique for correcting the process in light of lessons learnt, where process critical success factors are implicit but not identified, measured or controlled […] Historically in IBTech […] the PIR has not been done consistently. Lessons learnt have been anecdotal and qualitative, with low potential for determining the full set of opportunities for improvement. [IBTech intranet 10-06-2004, A8]

However, as was noted earlier in the chapter, not all corporate workers agree that measuring variations to the plan, identifying the causes of variation, and formulating 'lessons leaned' lead to process improvement:

> 'Lessons learned' aren't learned, they are just written down. They are in the head of the best people. PIR is a joke; there is just no time for that. A team must quickly attack a new project; there is no time for discussing metrics. Senior people possess a massive amount of knowledge. [Project manager, 11-05-2004]

6.2.4 Bureaucracy

In October 2004, a colourful and visually-rich pamphlet entitled "Our Operating Principles" was physically distributed to all employees. The document stated that the

executive management team had created a series of principles for employees to follow: "Our success relies on our collective ability to understand these principles and aspire to live by them everyday." According to the document, "operations should be fast and simple" and corporate workers should strive to "eliminate unnecessary bureaucracy" [Our Operating Principles, 2004].

The 'bureaucracy' discourse typically is about the elimination of activities that slow down the software development process and that do not contribute any value to end results. The following extract from a newsletter distributed electronically to all corporate workers outlines the importance of eliminating bureaucracy in order to become more efficient:

> Bureaucracy has real dollar costs. There's nothing worse than having to experience dumb policies and procedures […] It's important for managers to emphasize simplicity and efficiency as a guiding principle. Procedures often are put in place to accomplish a particular goal or address a particular issue, then continue to be used long after the issue becomes irrelevant. […] identifying day-to-day opportunities to eliminate bureaucracy is important in shaping an efficient culture. [Newsletter 26-05-2004, A99]

The development and maintenance of regional software development processes can be depicted as a bureaucratic activity:

> During the last two years, multiple codes of practices have been created in each region, in most cases, several per software house. This has allowed us to achieve a certain maturity corresponding to the CMM level 2. […] However, the development and documentation of a new COP each time a team wants to achieve CMM level maturity is highly time consuming and bureaucratic. [Newsletter 25-11-2004, A96]

In this frame, having a single software process to develop and maintain would appear less bureaucratic. Thus, institutionalizing a standard organizational software process can be said to contribute to eliminating unnecessary bureaucracy:

> The GreenBook is a central body of standards and practices that represents IBTech best practices […] If we are successful then our body of standards and practices will help those trying for CMM level 3 to get there more easily […] as we develop and implement efficient,

> rationalized working practices we eliminate unnecessary procedural duplication and bureaucracy. [Newsletter 25-11-2004, A96]

Following this line of reasoning, RUP, which was intended to provide IBTech with a standard software development process, should have contributed to eliminating unnecessary bureaucracy. However, as explained in the previous chapter, RUP was in the organization generally perceived as being too comprehensive and excessively document-centric. In September 2004, a newsletter discussed the limitation of the RUP platform overtly:

> The Equities Process Framework (EPF) is currently based on Rational Unified Process (RUP). RUP is comprehensive and designed to be configured by an organization but there is the opinion that it could be too heavyweight for the organizational needs. There is the thought that Agile practices could be applicable in this instance. [Newsletter 08-09-2004]

When asked if he believed that developers find the claim that RUP would reduce the non-value added activities credible, a project manager said: "My understanding – that will give rise to great hilarity among developers. I don't think they've got any great affection for RUP."

For some, however, there is nothing wrong with RUP: the problem resides in the way the platform is enacted. According to one corporate worker, a software organization has to strip RUP from unnecessary process and artifacts in order to reduce bureaucracy activities to an acceptable level:

> A cause of failure of RUP projects is the tendency of process-immature organizations like IBTech to examine RUP and say "There's a lot of good stuff here; this is perfect for us". These organizations then install the base RUP product and tell their staff to follow all of its 3,000+ HTML pages. There is a lot of good stuff in RUP, but we need only a small subset of it. [Email from CMM representative 26-05-2004, A13D]

The 'bureaucracy' discourse also surfaces in organizational texts about the PIR. The PIR is particularly vulnerable to anti-bureaucracy rhetoric because this process is not essential to the software development process. The PIR is criticized for being too bureaucratic, as the manager responsible to process confesses:

> I have even reduced the number of pages in our PIR report template to make it less intimidating. Many still see the PIR, and in particular PIR documentation, as an unnecessary overhead, even though there is little effort involved… it's a culture change we are pushing at the moment. [Email from manager responsible for the PIR 06-08-2004]

6.2.5 Communication

The 'communication' discourse surfaces in the organizational text entitled, 'Our Operating Principle.' The document previously introduced contains a section entirely devoted to communication. Under the heading "Communicate honestly, clearly and often," the document urges corporate workers to:

- Share ideas and information as quickly as possible
- Share your passion for learning and knowledge
- Reinforce important messages clearly in writing
- Communicate important message in person
- Be concise and keep it simple [Our Operating Principles, 2004]

The 'communication' discourse inhabits most organizational texts about agile development. The following extract from a newsletter discussing the benefits of agile illustrates this:

> The largest single implication to managers working in the agile manner is that much more emphasis is placed on people factors in the project […] the quality of communication become first-rate items of concern for the would-be agile team. [Newsletter 20-10-2004, A180]

In the same spirit, an internal report written by members of IBTech's agile working group identified communication as a critical success factor:

> Critical success factors: […] Communication: talking and engaging with customers in a collaborative fashion is very important. This should be maintained throughout the project, but is especially critical early on. [Internal report 24-09-2004]

The ‘communication’ discourse is also invoked when it comes time to sell agile development to the different software houses following an allegedly-successful pilot implementation. The following extract shows this:

> Using Agile has really streamlined MISTE’s [market interactive system for trading electronically] end-to-end project processes. MISTE now has a dynamic approach to gathering user requirements, estimating and scheduling releases, and monitoring and controlling project deliverables. Overall, it has […] established better communication with dependant AD [application development] teams. [Newsletter 21-10-2004, A81]

The ‘communication discourse’ also inhabits organizational text appertaining to RUP. A passage from an IBM Rational white paper found at IBTech is illustrative:

> Improved Communication: By using a proven methodology [the RUP methodology] and sharing a single comprehensive process, your team will be able to communicate more effectively and work more efficiently. [IBM Rational white paper 2001, A53D]

In all the excerpts above, ‘communication’ is understood along the lines of agile development-- that is to say, as face-to-face communication, not document-based communication. The ‘communication’ discourse can, nonetheless, occasionally be found in organizational text about the CMM/I, the PIR, and SEPIG. For example, a guide for software improvement obtained by IBTech from the SEI depicts communication as a tightly-managed activity involving the creation of a communication plan and formal meetings:

> Communication plans also need to be developed to insure that communication of the events associated with the SPI program are received properly by the organization. Each of these plans will have schedules that must be monitored and defined milestones that must be reviewed [p. 167]. Establish organization-wide communication vehicles (such as newsletters, town-hall type meetings, brown bag seminars) to keep the entire organization informed on the progress and results of the SPI program. The members of the organization should be periodically surveyed to ensure that the messages are being received. [p. 39]. (McFeeley, 1996)

6.2.6 Innovation

The 'innovation' discourse pervades the organizational document outlining IBTech's objectives for 2004. Objectives under the heading, "Be an innovative organization," include the followings:

- Institutionalize the Productivity and Innovation process to ensure a culture that encourages and rewards innovation
- Enable people to feel empowered to propose radical ideas
- Share ideas and market intelligence between groups
- Ensure knowledge about patent process is clear and people dedicate time to this agenda [A37]

The patent process referred to in the foregoing extract deals extensively with the subject of innovation:

> We need to take our passion for excellence and translate it into innovations that take our business to the next level. We encourage you to think critically about how we do business and how we can improve our processes and products. Innovation not only cuts costs, eliminates errors, reduces effort, and creates new opportunity: Innovation can also be the intellectual capital that attracts new clients and talented employees to our firm. [IB Technology patent program 15-11-2004]

The document also includes a quotation from a highly-ranked corporate worker emphasizing the importance of innovation for IBTech:

> We need 20% growth to stay competitive, and 10% more to move ahead. That extra 10% comes from innovation – good old-fashion product and process development. This has to be a top priority for the firm. [IBTech patent program document 15-11-2004]

The 'innovation' discourse inhabits organizational texts about agile development. In a lecture delivered at IBTech, Jobs, the agile and iterative development consultant, argued that software development is innovation development in these words: "Most software is not a mass manufacturing problem. Software development is new product development" [A24]. Another organizational text about agile development states that:

> Agile development combines creative teamwork with an intense focus on manoeuvrability […] JP Morgan's leadership is fueled by innovation – the new idea that become broadly adopted and has big economic impact. The ability to innovate effectively is a hallmark of the best organization. [Internal report 15-11-2004]

6.2.7 Learning

The notion of a 'learning organization' expounded by Peter Senge in the renowned book, "The Fifth Discipline," captivated many corporate workers. From Senge's original notion, corporate workers derived their own definition of a learning organization:

> An Organization that learns and encourages learning among its people. It promotes exchange of information between people hence creating a more knowledgeable workforce. This produces a very flexible organization where people will learn from experience and adapt to new ideas and changes through a shared vision. [Electronic presentation document 02-02-2004, A37]

The standard software process – at different points in time known as the GreenBook, the EPF, and RUP – aimed, among other things, at transforming IBTech into a learning organization:

> The GreenBook program has the goal of transforming IBTech Equities into a "learning organization", this being the only way to adapt, survive and succeed in an increasingly distributed, complex environment and a competitive, uncertain world. [Newsletter, May 2003]

The 'learning' discourse inhabiting organizational texts about the "GreenBook Program" was so forceful that the "GreenBook Program" was, in 2004, renamed the "Learning Organization Program." Traces of the 'learning' discourse can indeed be found in organizational texts about all components of the "Learning Organization Program' (see Table 6 in Chapter 5). The following extract shows that the 'learning' discourse inhabits organizational texts about SEPIG:

> The Software Engineering Process Improvement Group (SEPIG) is the forum to facilitate the definition, maintenance, improvement and establishment of the software engineering and management process for IBTech, allowing Areas for Improvement and Best Practices to be evaluated, and developing a culture of Continuous Process Improvement. SEPIG has a direct correlation with two IBTech Equities vision principles: Provide Excellent Solutions With Greatest Efficiency and Be a Learning Organization and Innovate. [Intranet 06-14-2004, A23]

Agile and iterative development place a considerable emphasis on learning. The following excerpt from the company intranet shows that the 'learning' discourse is present in organizational texts on agile and iterative development:

> Underpinning this approach [agile development] is the assumption that adapting to a fast and iterative cycle […] and learning to speed up our delivery and build on experience is critical. […] The cycles need to be short, so teams can learn from small rather than large mistakes. […] Iterative development and PIR are practices that expose results to scrutiny. [IBTech intranet 18-11-2004, A27]

Corporate workers describe the CMM/I in an electronic presentation document as a model that enables "better knowledge transfer and reuse." Moreover, the model is said in the same text to lead to the establishment of "more consistent processes" that enable the "internal transfer of lessons learned and best practices." The term 'learning' is not used explicitly, but it seems that the 'learning' discourse also inhabits organizational text on the CMM/I.

'Learning' is undoubtedly the most intricate discourse of all. This is so because 'learning' is highly interpretatively malleable and, therefore, may mean different things in different contexts. The next section discusses the tensions that exist within discourses.

6.3 Connections and contradictions

This section both concentrates on the connections among the discourses constituting software development in the organization, and highlights the contradictions within discourses. In Figure 3, discourses are represented by circles. A solid line denotes a strong connection between two discourses, and a dotted line, a very weak connection.

When applying the hermeneutic of suspicion, it becomes clear that a particular theme, a particular discourse, can be endowed with different meanings. In fact, a discourse can mean different things to different people.

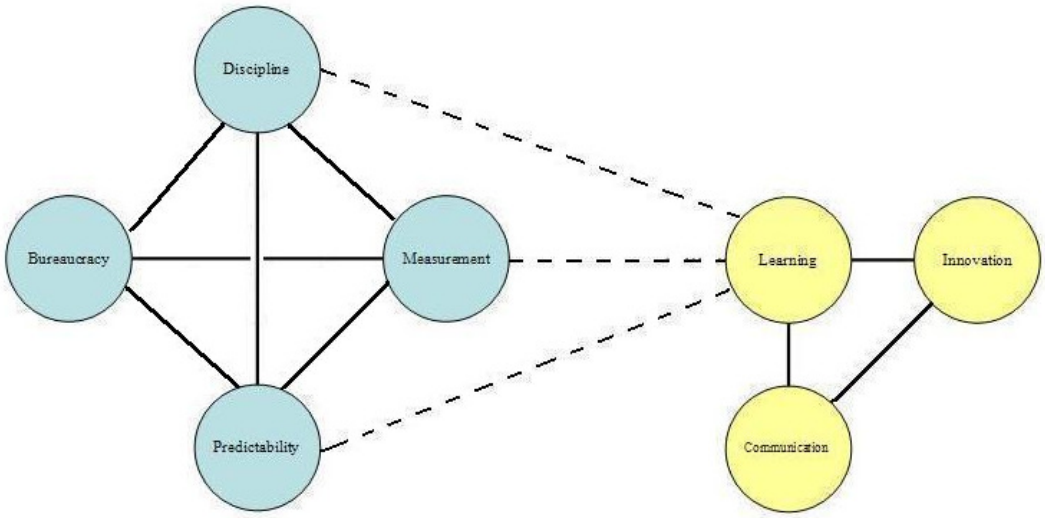


Figure 3: Connections among discourses

The frequency with which two or more discourses were found to be mutually supportive in the organizational texts analyzed was calculated and used as an indicator of the strength of the connection between two discourses. As such, discourses which were found together relatively often, and which supported one another, were identified as being mutually supportive. Table 9 shows the number of times that two discourses were found to be mutually supportive.

Table 9: Strength of the connection between discourses

	Discipline	Predictability	Measurement	Bureaucracy	Communication	Innovation	Learning
Discipline		54	33	45	2	0	8
Predictability			36	31	0	1	7
Measurement				24	0	0	7
Bureaucracy					0	0	4
Communication						14	17
Innovation							18

6.3.1 Discursive connection: Dominant IDF

The 'discipline' discourse is strongly associated with the 'predictability' discourse. Organizational texts analyzed imply that the more disciplined a process is, the more predictable the outcomes should be. The following extract from an IBM Rational white paper clearly shows the association between the 'discipline' discourse and the 'predictability' discourse:

> The Rational Unified Process or RUP product is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users with a predictable schedule and budget. [IBM-Rational white paper 2001, A22]

In the same spirit, the 'measurement' discourse connects harmoniously with the 'discipline' discourse and 'predictability' discourse. In the following quotations, the 'discipline' discourse, which inhabits the text segment "defined roles and responsibilities," dovetails with the 'predictability' and 'measurement' discourses:

> I can't overemphasize the importance of having a disciplined process because without it we can't make predictable deliveries. It is not that we are under intense pressure to periodically release software like Microsoft, but there is always room to further improve team efficiency and costs […] As far as predictability is concerned […] we have to have well defined roles and responsibilities, and with time, metrics to be measured against. [Interview 15-09-2004]

The CMM/I literature provides ample evidence that 'predictability,' 'discipline,' and 'measurement' are mutually-supportive discourses. This collection of quotations from the CMM/I reference book used at IBTech illustrates this fact:

> These [maturity] levels are a means of predicting the general outcomes of the next project undertaken. (Chrissis et al., 2003: 75)
>
> At maturity level 3, the standards, process descriptions, and procedures for a project are tailored from the organization's set of standard processes to suit a particular project or organizational unit and therefore are more consistent […]
>
> At maturity level 3, processes are managed more proactively using […] detailed measures of the process […]. At maturity level 4, the organization and projects establish quantitative objectives for quality and

> process performance […] Quality and process performance is understood in statistical terms […] detailed measures of process performance are collected and statistically analyzed. (Ibid: 80)

In the foregoing quotation, the concept of 'maturity level' is said to be used as a means of predicting the outcomes of software projects. The 'predictability' discourse thus seems to inhabit the concept of 'maturity level.' Moreover, the 'discipline' discourse appears to inhabit the notion of 'consistency,' which is reflected in an organization's maturity level. From this perspective, to be consistent is to be disciplined. Finally, the maturity level is an indicator of the extent to which the software process can be accurately measured.

Significantly, organizational texts analyzed imply that a process that is undisciplined and difficult to measure, and whose outcomes are unpredictable, is inefficient and ineffective. For example, a quality policy document states that "The purpose the Software Development Lifecycle shall be to […] adhere and comply with defined software development process to produce correct, consistent software products effectively and efficiently" [Internal report, A14]. Processes that do not comply with the standard development process are assumed to be inefficient and ineffective, and are labeled 'bureaucratic.' The following except demonstrates that the 'bureaucracy' discourse supports the 'discipline' and 'predictability' discourses:

> The GreenBook is a central body of standards and practices that represents IBTech best practices […] If we are successful then our body of standards and practices will help those trying for CMM level 3 to get there more easily […] as we develop and implement efficient, rationalized working practices we eliminate unnecessary procedural duplication and bureaucracy. [Newsletter 25-11-2004, A96]

However, organizational texts analyzed do not show the existence of strong links between 'discipline,' 'predictability,' 'measurement,' '(anti-)bureaucracy' and the other discourses found in the organization. These four discourses – and only these four discourses – work together to constitute a vision of software development.

In sum, four discourses work together to constitute a vision of software development: software development is an activity that is best when discipline, predictable, measurable, and efficient (non-bureaucratic).

6.3.2 Discursive connection: Alternative IDF

'Communication' and 'learning' are mutually-supportive discourses. For example, a connection between the 'communication' discourse and the 'learning' discourse is manifest in the memo announcing the launch of a wiki at IBTech:

> The idea is to create a way for technologists to form a community where they can ask questions, gain insights and share lessons learnt. […] If it is easy and they know others will use it [the wiki], people willingly reach out and share their information. The idea is to make communication and knowledge sharing so easy, that people just do it.
>
> One of IB Technology's goals is to be a learning organization. This [the wiki] is a bottom-up tool that works as a means to share knowledge at every level of the organization. [Memo 05-06-2004, A21]

In the foregoing excerpt, a link between the 'communication' discourse and the 'learning' discourse is made. Wikis, the argument goes, make communication easy and so foster learning.

A connection between the 'communication' discourse and the 'innovation' discourse is also apparent. The following extract highlights the importance of sound internal and external communication to innovate:

> Frequent and continuous communications, both within and outside the team and firm, are critical to the success of the development process […] The more frequent the communications, the more information can be transferred. Moreover, because of the uncertainty often associated with these projects, different types of innovation may be needed as more uncertainties are resolved. Thus, communication should not only be frequent, they should also be continuous over the life of the project […] [Internal report 06-08-2004]

However, despite frequent attempts, the 'learning' discourse was only linked to the 'measurement' discourse with great difficulty. From an engineering perspective,

learning depends on measuring variations to the plan. The quality of a process is measured in terms of how well it goes according to the plan. In this frame, the PIR and SEPIG could be said to enable learning by qualitatively and quantitatively identifying variations to the plan. The official description of the PIR makes this claim:

> PIR delivers little value without a systematic approach and technique for correcting the process in light of lessons learnt, where process critical success factors are implicit but not identified, measured, and controlled. PIR is a process review and a learning component of the software development model […] areas for improvement, lessons learned, best practices are used in the Software Engineering Process Improvement Group (SEPIG). [Intranet 23-03-2004. A8]

However, many see the notion of learning as being at odds with the engineering perspective, which emphasizes discipline, predictability, measurability, and efficient software development. For example, in the following verbal exchange, an agile consultant hired by IBTech rejects the connection between the 'measurement' discourse and the 'learning' discourse:

> Consultant: A key part of agile development that is now recognized – I think – is the notion of retrospective […] The one I've seen were iteration retrospectives, which would only take about one hour.
>
> Researcher: It's a kind of mini PIR, isn't?
>
> Consultant: Well, no. It's not a PIR because you are not trying to find out who was at fault - simply what worked well, what didn't work so well, what we could change. They are also done at a release level and take a day, two days.
>
> Researcher: Do you associate the PIR with finding the one who's at fault?
>
> Consultant: That's one of the key things that agile would not attempt to do. I think a very good example of a learning organization in practice is [company ABC]. It's one of the first ever XP teams and they had a complete commitment to continually try reflect on where they are, where they are going. They introduced a range of strategies that tried to recognize the role of the individual, despite the importance of the team. So they introduce the notion of 'gold card days'. Each developer was given two 'gold card days' each month. So they took explicit action.[9]

[9] A gold card allowed a developer to spend a day per month studying a technical subject of his or her choice.

> Those are the kinds of practical concern that allows that sort of group to say 'we are a leaning organization'. [Interview 14-12-2004]

In sum, the analysis has revealed that there is an association between the 'measurement' discourse and the 'learning' discourse, but that the association remains weak. The association is considered weak because the link between the two organizational discourses is seldom made, and when it is made, it is generally contested. The agile consultant's narrative, presented above, exemplifies this fact.

Strong associations exist between 'learning,' 'innovation' and 'communication.' These three organizational discourses are mutually-supportive and work together to constitute an alternative vision of software development. The vision of software development the three aforementioned discourses sustain is considered to be "alternative" (Fairclough, 1995: 12) because 'learning,' 'innovation' and 'communication' are less frequently present in organizational texts than those of the "dominant" IDF. Moreover, when present in organizational texts, the discourses generally serve to represent something novel that requires explanation, such as agile development, iterative development, or the use of wikis.

7 Analysis

This chapter illuminates how and why organizational actors at the study organization came to think about software development in certain ways and adopted particular practices. The chapter argues that developing an in-depth understanding of the relationship between the actors' interests and the social-organizational structures provides greater insights on the discursive constitution of software development. Bourdieu's concepts of 'field of struggles,' 'field of forces,' and 'capital' are employed to illuminate the relationship between, respectively, the professional struggle in which actors were engaged, the logic of the broader socio-institutional context, and the competences that actors possessed.

7.1 Field of struggles

All of what was done at IBTech was, in one way or another, related to the business of software development. It was the organization's mission to develop and maintain software systems, and corporate workers were expected to advance causes that directly or indirectly contributed to this mission. The responsibilities attached to an organizational role specified how an individual should contribute to the organization's mission.

In the process of fulfilling their responsibilities and enacting their professional roles, corporate workers supported beliefs about software development. For example, in completing a PIR, a project manager contributed to reaffirming the importance of process improvement and upheld the particular vision of software development that it implied. Alternatively, by repeatedly reporting the completion of a PIR, a project manager signalled that process improvement was less important than other concerns, or that the particular vision of software development that the PIR presupposed did not concur with what s/he believed software development was. So although the work of corporate workers took place within the boundaries set by their respective roles, corporate workers possessed a fair degree of discretion as to what they did within these boundaries. Consequently, corporate workers possessed a certain degree of discretion over the beliefs they chose to endorse through their practices.

Corporate workers tended to practice software development in a way that was consistent with their beliefs regarding what software development should be, and should involve. When asked to justify their practices, corporate workers generally invoked an "efficient resources allocation" argument. For example, when asked why an activity that should have been completed had not been completed, they would often claim that they had lacked the resources, or had chosen to use their resources elsewhere, in an attempt to give priority to more important activities. Seen this way, corporate workers were constantly engaged in confirming or contesting how software development should be thought of and practiced.

It is suggested here that the constitution of software development occurred in reality through mundane practices and was ongoing. Although the 'learning organization' program was supposed to provide the formal mechanisms to shape software development, its shaping was constantly occurring. For this reason, there is a strong argument for concentrating on the actions of individuals in situ – i.e., on their practices. At IBTech, the production, transmission, and interpretation of texts about software development was part and parcel of, and inseparable from, the activity of software development. From this perspective, what corporate workers wrote and said in the context of work was always intended to confirm or change what software was.

Significantly, findings do not suggest that the way in which software development was principally thought of and practiced had a major implication on corporate workers at a personal level. The degree of legitimacy that beliefs and practices were endowed with did not change anything as regards a corporate worker's salary, and had little bearing on his or her career prospect. Role and departmental boundaries limited the amount of influence that an individual could have on software development, and corporate workers were cognizant that their individual contribution to changing how software development is understood and practiced could only be minor. Moreover, it was quite common for individuals to occupy a position for only a limited number of years and then move on to another position outside IBTech within the investment bank. For these reasons, corporate workers generally did not regard the definition of software development as a professional project.

Because the manner in which software development was principally thought of and practiced was fairly inconsequential to corporate workers at a personal level, one would expect them not care about it. However, as participant observations suggested, corporate workers were expected to be interested in the organization's pursuits to the point of willingly subordinating their personal interests to those of the organization. The extent to which professionalism and a sense of duty was reflected in corporate workers' behaviour contributed to the elevation of their status. The definition of software development provided an opportunity to demonstrate an interest in what the organization did, and an ability to live by the organization's values.

The previous chapter opened with an insider account intended to give some sense of the authority structure that prevailed in the case study organization. It was noted that although corporate workers showed a remarkable status consciousness, status must not be understood simplistically in terms of rank in the formal hierarchy. It was observed that status was primarily determined by the recognition one received from others. Recognition was gained by showing commitment to the organization by adhering to its symbolic universe. More specifically, this entailed adopting the lifestyle that the corporation promoted and placing the interest of the corporation in front of one's own interest. At IBTech, producing such patterns of behaviour was understood as professionalism.[10] One's position in the formal hierarchy was, therefore, more a consequence of the recognition received than a source recognition and authority valid outside the vertical lines of the communication system of the formal hierarchy.

Participant observations also drew attention to the strong desire of corporate workers to elevate their status. Although this desire may be a human trait observable in most professional environments, it should not be too rapidly dismissed as being analytically irrelevant. At IBTech, a remarkably strong desire to improve one's status provided an impetus for the enactment of discursive and non-discursive practices intended to communicate professionalism. For this reason, it is appropriate to see the

[10] This treatment of the concept of professionalism is consistent with the literature. For Roberts (2005), professionalism refers to the extent to which one accepts the values of a role or a profession, and can be evidenced by an individual's ability to meet the normative expectation of a role or a profession.

discursive practices produced as part of the activity software development as practices capable of communicating professionalism. This proposition will be refined as the analysis progresses.

It is important to note that the proposition does imply that corporate workers produce texts with the conscious intent of communicating professionalism and improving their position in the organization. The discursive practices of corporate workers seem to escape any conscious strategizing and cause-effect calculation. As will be shown in details in this chapter, corporate workers possessed a distinctive set of norms that inclined them to talk about software development in a particular way.

Corporate workers shared distinctive norms which defined them as a cohesive group. Central to these norms were the importance granted to status, and the link, in the corporate workers' minds, between professionalism and status. The noteworthy desire to elevate their status ensured their commitment in the constitution of software development. Corporate workers were generally not inclined to critically question the basis for status determination, but accepted the pursuit of a higher status as a matter of course. Moreover, they agreed that the stakes were worth struggling for, even if their chances of success were low compared to those of other individuals.

The continuation of the struggle required an unconditional investment and an unquestioning belief in the struggle. To challenge the value of status and the legitimacy of the factors that determined it would have meant that the struggle was not acceptable as it was. Hence, the conduct of the struggle presupposed a fundamental accord or complicity on the part of corporate workers.

Of course, the corporate workers could, in theory, have refused to conduct themselves in accordance with the normative standards of their professional group. For example, they could have been cynical regarding the level of bureaucracy, and openly questioned the technical merits of established ways of doing things. More specifically, they could have very well rejected bureaucracy as a means of coordination and ignored the vertical communication channels provided by the hierarchy. However, corporate workers, who were status conscious, had an interest in

behaving to the best of their ability, for opting out of the struggle means abandoning one's status. As Bourdieu observes:

> The only absolute freedom the game leaves is freedom to withdraw from the game, by an heroic renunciation which – unless one manages to set up another game – secures tranquility only at the cost of social death (Bourdieu, 1981: 316)

Some discourses at IBTech were valued more highly than others, and the way one demonstrated competency was by producing texts that were highly valued. Using the notion of capital, the next section explains how the competence to create and use texts to constitute the object 'software development' authoritatively functioned as a marker of social status.

7.2 Capital

For Bourdieu, an analysis in terms of field necessarily involves analysing the "structure of the relations between the positions occupied by the agents […] who compete for the legitimate form of specific authority" (Bourdieu and Wacquant, 1989: 40). The form of authority is what Bourdieu also referred to as 'capital.' Analytically, the position of agents within the field is determined by the volume of capital possessed and the relative weight that the field grants to the different forms of capital.

Let us begin by examining what the forms of capital active at IBTech were, and how they structured the position of corporate workers. 'Position' should here be understood in two ways: as the stance of agents regarding how to think of and practice software development, and as the position occupied by agents in the field (e.g., dominant or dominated). As should be clear to the reader by now, the theory of symbolic violence is founded on the idea that there is a correspondence between the stance of agents and the position determined by the volume and forms of capital possessed.

Corporate workers communicated through a number of practices that they in effect adopted the lifestyle that the corporation promoted and placed the interests of the

corporation in front of one's own interests. At the risk of sounding cliché, this point was illustrated in the previous chapter by drawing attention to a number of non-discursive practices that corporate workers engaged in, including the choice of dress, hairstyle, and entertainment. Wearing a necktie, sporting business look hairstyle, leaving the office late, and playing golf at weekends were instances of practices that evoked professionalism and called for recognition. In a similar vein, it was advanced that organizationally-sanctioned consumption was also a practice that communicated an understanding of what the organization valued, as well as a willingness to meet the expectation of the organization. For example, the professionalism that RUP radiated weighted heavily on the decision to acquire the process.

More significantly, however, the discursive practices of organizational actors were not only practices capable of transmitting a message between a sender and a receiver, but were also practices that communicated an understanding of what the organization expected, and a willingness to meet its expectations. In this sense, discursive practices, just like the non-discursive practices referred to above, are practices capable of signalling one's degree of professionalism and, hence, one's status. Bourdieu nicely expresses this point in relation to the concept of capital as follows: "Every linguistic exchange contains the potentiality of an act of power, and all the more so when it involves agents who occupy asymmetric positions in the distribution of the relevant capital" (Bourdieu, 1992: 145).

Seen this way, the discourses that corporate workers drew on in their routine activities were not only capable of expressing their position on relevant software development issues, but also indicated these workers' positions in the social order. The capacity to use and create texts appropriately in the organization (i.e. professionally) had the effect of signalling where one stood, and served as a resource to maintain or elevate one's status. For this reason, it is appropriate here to speak in terms of 'linguistic capital' (Bourdieu, 1991: 66).

As previously explained in the theory chapter of this doctoral thesis, linguistic capital is the capacity, or competence, to create and use texts to constitute a social object compellingly and, thereby, change or reaffirm the practices of individuals in relation to the social object. It is important to note, however, that the notion of linguistic

capital goes beyond the purely linguistic aspect of language and represents a practical understanding of the norms regulating discursive practices. The next section will provide justification for employing this notion in order to understand why software development was constituted as it was at IBTech.

7.2.1 Justifying the use of the concept of 'linguistic capital'

Before deploying analytically the notion of 'linguistic capital,' it is necessary to justify why it is better suited than other more conventional notions of authority. In particular, it is appropriate to examining why the authority derived from the occupancy of a post (bureaucratic authority) and technical expertise (professional authority) could not be used by organizational actors to shape software development and could not be considered here as active forms of capital.

We recall that for Bourdieu, the notion of capital plays a dual analytical role. First, capital functions as a weapon of struggle. Furthermore, capital is a marker of social status. Consequently, in order to identify what the forms of capital active in the field were, two questions were probed: What resources were drawn upon to establish the legitimacy of beliefs about software development and constitute software development practices? What determined the status of individuals at IBTech?

Findings suggest that bureaucratic authority was largely inadequate for establishing the legitimacy of beliefs about software development. Bureaucratic authority gives an office holder the right to issue commands and expect obedience from his or her subordinates (Weber, 1968), but it cannot be used to make people believe on demand that one particular software development practice is superior to another. When it comes to changing beliefs in an economic organization, unobtrusive and normative mechanisms are generally deemed more effective (Jermier, 1998; Sewell, 1998). Bureaucratic authority was, nonetheless, used at IBTech to retain control over the choice and use of instruments of normative control (e.g., the 'learning organization' model).

Regarding the constitution of software development practices, bureaucratic authority was not a resource drawn upon by corporate workers. This is primarily because a

bottom-top approach was utilized to constitute software development practices. With the 'learning organization' governance structure, areas for improvement and best practices were identified and communicated by means of a PIR report completed by project members. There was an attempt to impose a standard software development process "from above" with the GreenBook, but it resulted in a humiliating disaster.

The possession of technical expertise, in the sense of technological proficiency, appears marginal in the constitution of beliefs and practices. To be sure, a certain degree of technical expertise was required to meet the requirement of an organizational role. However, findings do not suggest that technical expertise determined the capacity of a corporate worker to establish beliefs about software development and constitute software development practices. As a result, technical expertise cannot be a relevant form of capital.

The second question ("What determined the status of individuals at IBTech?") is appropriate because capital, by definition, determines who occupies the dominant status within a field of interaction. Participant observation data indicated that the status of individuals was not determined by the hierarchical rank they occupied. Despite of IBTech's traditional hierarchical organizational structure, status was largely determined by the recognition one received from others. A high hierarchical ranking and the possession of a significant volume of bureaucratic authority were consequences of the recognition one had received, but not the main source of recognition. A corporate worker had to be known to adhere to the norms promoted by the firm, consistently and over time, in order to be promoted. Consequently, the bureaucratic authority granted by a formal hierarchical rank cannot be considered a form of capital.

To reiterate a point previously made, recognition from others was principally obtained by communicating an allegiance to the norms of the organization through a wide array of practices. For example, commitment to the norms of the corporation was expressed non-discursively by leaving the office late and discursively by speaking about software development in a way that was deemed appropriate. Corporate workers that were capable of showing that they adhered to the values of their organization tended to receive recognition from others and a higher status. On

the other hand, those who did not understand what their organization expected from them, because of a lack of aptitude or experience, or who showed themselves to be discursively hostile to what working for an investment bank involved, were generally condemned to an 'unprofessional' social identity and a lower status.

7.2.2 Linguistic capital

The notion of linguistic capital, as hinted above, implies the capacity to produce texts that are effective. In order to speak effectively, individuals must make the texts they produced concur, to some extent, with the demand of the field in which they are situated. In a given field, some discourses are valued more highly than others, and part of the practical competence of individuals involves knowing how and being able to produce utterances that are highly valued in the field concerned.

In the study organization, linguistic capital not only operated at the sender's end; linguistic capital was also the competence that enabled the readers and hearers of a text to recognize its meaning and value. Thus, the effectiveness and consequentiality of a text not only depended on the capacity of the sender to make his/her product conform to the orthodoxy of the field, but also depended on the capacity of the receiver of the text to let meaning emerge from it, appropriate it, and allow the text to become consequential. Bourdieu captures this idea by asserting that "agents [are] endowed with schemes of perception and appreciation" (Bourdieu, 1991: 39).

Linguistic capital is a powerful notion that goes beyond the conventional treatments of authority. In the present case, bureaucratic authority and professional authority did not do adequate justice to the complexity of the phenomenon under study. Linguistic capital functioned as a form of authority that enabled corporate workers to produce texts that were consequential and to use them effectively in the definition of software development.

However, linguistic capital was not a 'magic powder' that could make everything written or said consequential. The fact that a corporate worker possessed a significant volume of linguistic capital did not mean that any texts he or she produced were automatically consequential. Linguistic capital is a capacity that enables individuals

to produce texts that meet the criteria of acceptability of the field. To put the matter differently, it is a capacity to communicate what the field, and in those engaged in it, value and believe to be right. For example, corporate workers at IBTech generally valued what was disciplined and predictable within the context of work. Consequently, texts depicting software development as a disciplined and predictable activity tended to be received favourably in this organization. The possessor of a significant volume of linguistic capital knew (at least implicitly) that depicting software development as an activity that could not be made predictable due the emerging nature of the requirements could only be done at the risk of being frowned upon.

One of the key ideas of the previous section is that there was at IBTech a correspondence between the definition of software development and the struggle for status elevation. The definition of software development essentially provided the platform for reasserting and negotiating statuses. In other words, statuses were reasserted and negotiated in acts of communication occurring in the process of negotiating how software development should be thought of and practiced. Seen this way, every act of communication became an opportunity to display the possession of linguistic capital.

At IBTech, utterances formulated within the context of software development – although the utterances needed not be about software development – provided cues as to "who was in and who was out," and "who had it and who did not." Based on those cues marking the practical competence of corporate workers, the social identity and self-image of individuals was established. This observation concurs with Bourdieu's assertion that linguistic capital serves as a marker of identity and status. For him, the "sense of the value of one's own linguistic product is a fundamental dimension of the sense of knowing the place which one occupies in the social space" (Bourdieu, 1991: 82).

However, the term 'linguistic' is somewhat misleading. In common usage, 'linguistic' refers to the knowledge of words and the rules for combining them. From a strictly linguistic point of view, as Bourdieu colorfully remarked, "anyone can say anything and the private can order his captain to clean the latrines" (Bourdieu, 1991:

191). Because of the problems that the use of the term 'linguistic' poses, the notion of linguistic capital will have to be refined later in this chapter to account for the distinctive language on which it is based. 'Language,' in addition to signifying a code that establishes equivalence between symbols and meanings, means "a system of norms regulating linguistic practices" (Bourdieu, 1991: 45). A language is a system of norms that provides a proficient language user a sense of what can and cannot be said. Languages, thus,

> imply a certain propensity to speak and say determinate things (the expressive interest) and a certain capacity to speak, which involves both the linguistic capacity to generate and infinite number of grammatically discourses, and the social capacity to use this competence adequately in a determinate situation. One the other hand, there are structures of the linguistic market, which impose themselves as a system of specific sanctions and censorships. (Bourdieu, 1991: 37)

To view a language as a system of norms regulating practices is consistent with the description of the field of struggles offered in the previous section. Corporate workers who abided by the demands of the corporation had to learn to speak in a certain manner and say determined things. Linguistic capital thus functioned as a practical mastery of a language that enabled the effective production of texts and utilization of discourses in the definition of software development.

To sum up, linguistic capital is the competence that enabled corporate workers to use a language to produce texts that were likely to be judged acceptable. Linguistic capital also enabled corporate workers to distinguish the texts that were acceptable from those that were not. All other things being equal, the more capital one possessed, the greater his or her ability to discursively influence the object of interest and, thereby, change and reaffirm the practices of individuals in relation to that object. The next section examines the language that was principally used at IBTech to talk about software development and valorise the texts produced and interpreted.

7.2.3 The corporate language

It is proposed that to understand the discursive constitution of software development, it is necessary to pay attention to the unequal capacity of corporate workers to exploit

the discourses that were valued in the context in which they were situated. When producing texts about software development, corporate workers, who were generally eager to improve their status, tended to draw on discourses that made them appear professional and that were likely to provide them the approval of others. The closing part of this section takes a closer look at the particular language that was predominantly used at IBTech and that served as a standard to valorise the texts that corporate workers produced and disseminated. In so doing, the following pages seek to clarify what mode of expression was used to communicate that one abided by the firm's normative expectations.

In the present context, 'language' refers to a way of perceiving and talking shared by a community, a system of norms, rather than knowledge of individual words and the rules for combining them. The extent to which the majority of corporate workers had internalized the norms of the corporate world was observable in a plethora of non-discursive practices. Significantly, the internalization of the corporate way of life was also reflected in a shared way of perceiving the objects of the social world and representing them through language. Consequently, it is appropriate to think of linguistic capital as relating to a mastery of the corporate language.

The corporate language is seen here a system of norms reflecting the way of seeing and way of talking that the corporate context upholds. Within the study organization, a practical competence in the use of this language represented a form of capital because it enabled corporate workers to actively engage in the constitution and maintenance of a representation of software development. Seldom was software development represented in purely technical or analytical terms as computer scientists and social scientists, respectively, tend to represent it. Indeed, at IBTech, it was corporate language that was favoured over any other language when it came to speaking about software development.

Moreover, because the degree of mastery of the corporate language signalled the degree to which the norms of the corporation inhabited an individual, a practical competence in the use of the corporate language functioned as the marker of social status *par excellence*. The possession of outstanding technical competencies mattered little in comparison to a practical mastery of the logic and rhetoric that prevailed in

the corporate context. Seen this way, use of the corporate language demonstrated the possession of a legitimate competence (i.e. the ability to use to use the corporate language with ease) and contributed to reasserting or elevating one's status (Bourdieu, 1991: 55).

Discursive practices expressing the possession of linguistic capital provided an indication of the capacity of an individual to meet the expectations of the field. And since the perceived capacity of an individual to meet the firm's expectations principally determined the status of this individual, the importance of demonstrating a mastery of the corporate language becomes clear. Every discursive practice provided some signs as to where the individual was located in the social order. However, this does not mean that texts at IBTech were primarily produced, disseminated, and consumed in order to elevate oneself in the social order. Conspicuous acts of professionalism were reported in the previous chapter, but texts were generally not produced with the calculated intent of elevating status. On this point, the researcher could not agree more with Bourdieu, who believes that all practices function as signs of distinction, although they are generally not primarily intended to be so. Bourdieu observes:

> all practice […] is distinctive, whether or not it was inspired by the desire to get oneself noticed, to make one self conspicuous, to distinguish oneself or to act with distinction. Hence every practice is bound to function as a distinctive sign and, when the difference is recognized, legitimate and approved, as a sign of distinction. […] The pursuit of distinction – which may be expressed in ways of speaking […] – produces separations which are meant to be perceived or, more precisely, known and recognized as legitimate differences. (Bourdieu, 1991: 237-238)

7.3 Field of forces

So far the analysis has centred on the struggle waged at IBTech and the resources that corporate workers brought to bear on this struggle. The attention will now be directed towards the wider institutional context in which this struggle took place. This stage of the analysis is critical as it here that the correspondence between the micro and the macro is revealed. Using Bourdieu's concept of field of forces, this section endeavours to demonstrate how the socio-institutional context structured

what counted as a legitimate way of thinking of and practicing software development.

Before delving into the argument, it is necessary to expound how the field of forces is analytically constituted. The field of forces gives a particular form of capital its value. Consequently, the point at which its value evaporates marks the borders of the field of forces (Bourdieu, 1992: 198-199). Common sense suggests that the social ability to use the corporate language adequately is principally useful within the corporate context. Other languages may prevail in other contexts (e.g. medical research and the military) and be deemed appropriate to talking about software development. Because it is presumably in the corporate context that the mastery of the corporate language functions as a form of capital, it is appropriate to analytically consider this socio-institutional context as the field of forces.

Following Fairclough (1995: 36-42), an institution was previously defined as an apparatus of verbal interaction that sustains particular ways of talking based on particular ways of seeing (see Table 2 in Chapter 3). From this discourse analytic perspective, an institution is an entity that possesses its own norms of language use. It is proposed that the corporation can be considered as an institution. More specifically, the corporation is seen here as a normative institution that provides its members the norms needed to produce, interpret, and valorise texts. For the sake of clarity, this institution will henceforth be referred to with the use of a capital "C" (i.e. "the Corporation").

The use of Bourdieu's terminology has introduced a difficulty which can now be dissolved. This difficulty is of a terminological nature, and in part due to the translation of the author's work from French to English. Conceiving the mastery of the corporate language as linguistic capital has the effect of implying that the efficacy of the texts one produces depends on one's linguistic abilities. The term 'linguistic' poses problem because language does not, in this study, function in a purely linguistic sense (e.g. see Bourdieu, 1991: 44). Language implies a way of seeing the objects of the social world, an understanding of when to speak and when to be quiet, a way of speaking (including the correct choice of analogies), an understanding of a symbolic world, a bearing, an attitude, and other signs that are

noticeable in the discursive practices of individuals (Bourdieu, 1991: 123-124). Since the particular form capital the researcher has in mind was produced by the corporate context and indicated a mastery of the corporate language, it will hereafter be referred to as 'corporate capital' rather than 'linguistic capital.'

7.3.1 The Corporation as a normative institution

This section brings to light the intricate way in which the socio-institutional context structured what came to be seen as an acceptable way to talk about and practice software development. Particular attention is placed on the harmony that existed between the corporate workers, their practices, and what they did and what the Corporation asked them to do-- or, in Bourdieu's terms, on the "harmony […] between their [the agents] subjective 'vocation' (what they felt 'made' for) and their objective 'mission' (what was expected of them)."

The correspondence between the position of corporate workers in the field of struggles and the expectations of the field of forces is vividly exemplified in the profile corporate workers made accessible in the employee directory. Here is an example:

> Jim began his financial industry career in 1986 at Chase Manhattan Bank as a clerk in Check Processing. He held several different roles involving Corporate Data Center management responsibilities in operation management until 1993 where he joined JP Morgan. During his tenure at JPM till 1997 Jim managed FX Option Middle Office Operation and was the Business Manager for Emerging Market Front Office. Jim experienced the merger between Citibank and Travelers and held responsibilities which included Head of Asia Pacific Trading and Capital Markets Operations, Business Manager for Trade Operations and more recently Business Manager of North America Equities Technology. He is currently responsible for Financial Control, Quality control, Compliance, Governance and IT Management for the region. […] Jim holds an MBA from Columbia Business School. [IBTech intranet, 12-07-2004]

This professional biography, complete with a photo of Jim, open-collared and smiling happily (not provided here), expresses the harmony between what the corporate worker has been doing and what the corporate world expects from its employees. Although Jim started as a clerk in check processing, an area seen as

being particularly unglamorous, he improved his position in the field in remarkable ways by making his life concur with the expectations of the Corporation for almost two decades. The professional profile presents an individual who has showed allegiance to the corporate world and the way of life it presupposes, and who has gained a practical understanding of its functioning. For Jim, such an understanding has represented an asset, something to be proud of. For Jim, the corporate world has become home territory.

This professional profile not only presents the success of a single individual, but also provides a model to emulate. The text subtly communicates the kind of behaviour that is expected from a corporate worker. The text presents an ideal through which success can be achieved by adopting the corporate way of life as one's own way of life, and by placing the interests of the Corporation in front of one's own interests. Jim, like may others at IBTech, is presented as living proof that individuals disposed to enact the behavior (discursive and non-discursive practices) that is valued in this particular socio-institutional context tend to rise to the top of the pile.

More significantly, the text exemplifies a natural ability to fulfil one's job well, thanks to the fit between dispositions (of the agent) and expectations (of the field). In Bourdieu's words:

> This harmony [between disposition and expectation] may be expressed in their sense of being 'at home' in what they are doing, of doing what they have to do and doing it happily, or with a resigned conviction that they cannot do anything else, which is another way, though less happy one, of feeling 'made' for one's job. (Bourdieu, 1981: 308)

At IBTech, the harmonious correspondence between what was expected from corporate workers and their struggle for distinction was also manifest in texts produced by individuals whose experience of the corporate world was relatively recent. In a document intended to explain why a promotion was deserved, an individual in his early thirties wanting to become vice-president expressed the reasons that he feels himself to be the perfect candidate for the job, referring to himself in the third person:

> John has established credibility and strong working partnership with technologists […] This level of partnership is evidenced by the comfort shown in these external groups such as Legal Sourcing in allowing John to lead and manage complex and high priority sourcing engagement for the Bank […] This type of involvement is indicative of John's level of commitment and of his attitude to his professional career and life in general, setting a positive role model for those around him. [John D., 30-07-2004]

In this quotation, the corporate worker seeks to convince those who have the authority to promote him that he possesses the norms required to be a vice-president. In order to show that he possesses these norms, John invokes his level of comfort with working with individuals who were apparently inculcated the prevailing norms. By blurring the distinction between his professional and private life – "his attitude to his professional career and life in general" – John alleges that there is perfect harmony between his subjective 'vocation' (what he feels made for) and his objective 'mission' (what the corporate world expects of him). In other words, John's struggle for recognition involves acquiring the norms of the Corporation and communicating to others that he possesses them.

The correspondence between the socio-institutional position of individuals, their norms, and the expectation of the Corporation was also noticeable in several texts about software development. These texts suggest that their authors adhere strongly to the corporate logic that prevails at IBTech. Let us consider Eric's email reply to a colleague's follow up on an artefact for the RUP process-platform:

> Eric,
>
> I just wanted to touch base and see if you can send me updated relevant artifacts for the RUP platform. I am sure you are aware that the Learning Organization Transformation is a high priority.
>
> Vince [Email, 01-11-2004]

> Vince,
>
> I have been working on a number of significant matters the last few weeks for [the CTO] and unfortunately these have priority. I now have some assistance of a BIM [a recruit] who will be working on this for me so I am hoping to have revised content available for you next week. I am consistently working 65 hour weeks as it is so it's not a question of me simply ignoring this work but of limited bandwidth, until recently there

> has only be one of me covering 3 lines of business globally which is really a 3 person job.
>
> [Email, 02-11-2004]

Eric makes clear in his reply to Vince that he worked directly with the CTO. The researcher interprets this statement as an attempt to establish an ideational proximity with a high-status individual, which suggests status consciousness on the part of Eric. Moreover, in the original text, Eric referred to the CTO by his Christian name to emphasize his proximity and his ease in working with people who possess a significant volume of corporate capital. The message that Eric wants to communicate is that he possesses the norms that are valued. The second sentence makes clear that Eric is now someone's boss, which shows allegiance to the corporate way of doing things. The sentence suggests respect for the formal hierarchy, which indicates respect for the corporate way of life. Finally, Eric takes the opportunity to mention that he consistently works 65 hours per week. Working long hours is a means commonly used to communicate that one meets the expectations of the Corporation, and that one places the interests of the Corporation in front of one's own interests.

It is important to note that the three previous texts presented were not consciously intended to display professionalism or serve as ammunition in a struggle for status conservation and elevation. That is to say, the texts were not purposefully formulated in order to express the adoption of a worldview (i.e. a collection of beliefs about life and work) or the possession of the norms of language use of the Corporation. Rather, the texts were formulated in the de facto language without any apparent strategizing. Thus, the acceptance of the norms of the Corporation, including the norms of language use, was inscribed in the corporate workers way of being. This interpretation is commensurate with Bourdieu's observation that the acceptance of a language as legitimate is normally inscribed in the dispositions and practices of actors:

> The recognition of the legitimacy of the official language has nothing in common with an […] intentional act of accepting the 'norm'. It is inscribed, in a practical state, in disposition which are impalpably inculcated, through a long and slow process of acquisition, by the sanctions of the linguistic market [i.e. the field], which are therefore adjusted, without any cynical calculation or consciously experience constraint, to the chances of material and symbolic profit which the laws

> of price formation characteristic of a given market objectively offer to the holders of a given linguistic capital. (Bourdieu, 1991: 51)

In sum, there existed a harmony between the corporate workers and their practices and what the Corporation asked them to do. Being inculcated with the dominant normative understanding of work and life renders individuals sympathetic (or antipathetic) to some discourses and practices. Let us now transpose this argument to the case of the discursive constitution of software development.

7.3.2 Field of forces and software development

Corporate workers were inculcated, through immersion in the corporate context, with the norms prevailing in this context (including the norms of language use). It is the possession of these norms that defined corporate workers as a cohesive group. An important aspect of these norms consisted of the remarkable significance granted to status and the manner in which discourses were valued.

The norms that corporate workers possessed were reflected in language use. These norms predisposed them to use the language of the Corporation, rather than any other languages, to produce and interpret texts about software development. These norms also made them sympathetic to certain discourses. Because corporate workers had to be perceived as being professional in order to maintain and improve their status, they tended to use the language that was most likely to make them look professional when producing and interpreting texts about software development. This language was the language of the Corporation.

It should be stressed once more that from the analytic perspective adopted, the Corporation sustained not only a way of talking, but also a way of seeing the objects of the organizational world. The institution (the Corporation) privileged certain discourses over others, which were used to represent the objects of the organizational world (Bourdieu, 1991: 67 & 169; Fairclough, 1995: 37-41). Consequently, the institution structured how the objects were discursively constituted. In the case of software development more specifically, findings suggest that the Corporation put a premium on discourses of ‘discipline,’ ‘predictability,’ ‘measurement,’ and ‘(anti-)

bureaucracy.' These four discourses formed the object 'software development' that was 'dominant' at IBTech.

The corporate language dominated the texts that individuals at IBTech encountered as part of their routine daily activities. This language often concerned itself with the effective and efficient pursuit of an instrumental goal. This goal needed not be software development per se. For example, texts formulated in the corporate language were often about managing human and material resources, employing techniques designed to increase the efficiency of outputs, and affecting organizational changes effectively. What made the corporate language distinctive was that it tended to represent an activity as an instrumental activity and, what is more, an activity based on considerations of efficiency and calculation. Almost everything surrounding the an activity became seen as "business problematic" (Swanson and Ramiller, 1997).

However, although the texts that circulated within corporate field were generally not produced with software development in mind, they were, in the present case, received in an organization whose mission was to develop software systems and to manage software projects. The discourses those texts conveyed composed the stock of discourses of the organization. In other words, they were the "discursive resources" (Hardy et al., 2000) available to corporate workers to take positions on issues involving software development.

For example, the 'learning' discourse, which is a discourse accepted within the corporate field (Contu et al., 2003), suggests that the know-how of individuals has to be managed in a way that helps to achieve specific outcomes. The general 'learning' discourse was, at IBTech, tied to software development and utilized to represent it. The 'learning' discourse was utilized, among other things, to promote organizational goals such as improving the firm's capacity to innovate and transfer internal best practices across regions.

Using the corporate language to formulate and interpret texts about software development induced corporate workers to draw on discourses that the Corporation celebrated. As if by magic and without any apparent effort on the part of corporate

workers, the use of corporate language tended to help these workers adjust their discursive products to the expectations of the Corporation.

It was in the practices of the text production and interpretation that the symbiosis between the corporate worker and his position in the field, on the one hand, and the Corporation, on the other, was most apparent. One was not a corporate worker until one had demonstrated possession of a certain volume of corporate capital. Until one's ability to utilize the language of the Corporation was known, one remained an outsider, excluded from the social-institutional context in which this form of capital was created. The desire to conserve the capital acquired through previous struggle led corporate workers to utilize the language of the Corporation – to enact the Corporation (as an apparatus of verbal interaction) – when taking a position on software development. In so doing, his position in the field (*qua* corporate worker) was made to converge with his position on software development (that of a 'genuine' corporate worker). There was nothing the corporate worker could do to advance his position that did not, *ipso facto,* served the Corporation.

Following this line of reasoning, it would be a mistake to try to understand how software development was discursively constituted in terms of the immanent logic of the field of forces (i.e. the macro), just as it would be a mistake to try to account for it exclusively in terms of the individuals' interactions and mundane practices enacted as part of an ongoing status competition (i.e. the micro). Software development was the offshoot of two compounding forces (micro and macro). Software development was formed through the repeated encounters of individuals with an institution – by the struggles of individuals to appropriate an institution by using its language.

It is important to stress that the corporate worker did not demonstrate a conscious desire to promote the interests of the Corporation. It just happened, thanks to the correspondence between the corporate worker's desire to elevate himself in the social order (by accumulating the capital which enabled him to do so) and the norms that were inculcated through immersion and which served him of capital. The corporate worker was totally identified with his position in the field of forces – to the point that it would be impossible to try to determine which of his practices in the corporate context were the product of his volition, and which were those of the Corporation.

The Corporation functioned as a apparatus which, once set in motion, produced enough energy of its own to remain in motion. Corporate workers, once they had acquired the norms of the socio-institutional context, had no choice but to seek to maintain or elevate their social status-- i.e. to conserve or increase the specific capital that was only created within the field. Corporate life presupposed a genuine interest in the struggles in which corporate workers engaged. To refuse to produce accepted statements, which was never an option consciously considered by a sufficiently socialized individual, would have meant falling into oblivion (Bourdieu, 1991: 316). The corporate worker had an acute understanding of what was expected of him in terms of discursive practices. He was in harmony with what was expected of him. This harmony was expressed in his talking about software development in the language of the Corporation with the conviction that this was how one should rationally talk about software development.

7.3.3 Double vision of a single object

Focusing on the influence of the socio-institutional context, it was argued above that the Corporation imposed its own particular logic on its corporate workers in such a way that these corporate workers and the Corporation became one. From this perspective, the interests of the Corporation and those of its members became inseparable: there was nothing that they could do to advance their own interests that did not, by the same token, help defend the interests of the Corporation.

However, if the Corporation imposed its logic so forcefully on its members, how, then, can one explain the existence of two different discursive ensembles – one 'dominant,' the other 'alternative'? These discursive ensembles sustained different ways of thinking about software development and presupposed different software development practices. Does not the existence of two different ways of thinking about and practicing software development at IBTech prove that the Corporation had only a limited capacity to wield power -- hence proving that resistance to its logic was alive and well?

Fairclough notes that "it is generally possible to identify a 'dominant' IDF and one or more 'dominated' IDFs in a social institution" (Fairclough, 1995: 41). From this perspective, the struggles between the different factions within the institution centre upon keeping a 'dominant' IDF dominant, or challenging this dominant IDF in order to replace it. Fairclough contends that it is when the dominance of an IDF is unchallenged that the practices it presupposes become most naturalized.

Seen this way, the 'dominant' discursive ensemble at IBTech should, in theory, have reflected the orthodoxy of the Corporation, and the 'alternative' discursive ensemble, challenged it. Thus, the ideal of the Corporation should have been quite commensurate with the discourses of the 'dominant' ensemble, but should have clashed with those of the 'alternative' ensemble. The discourses of 'learning,' 'innovation,' and 'communication' should have represented a source of resistance to the orthodoxy of the Corporation.

Here, the researcher's interpretation of the findings does not concur with the theoretical propositions laid out by Bourdieu and Fairclough. In the study organization, the software development practices that were most naturalized leaned towards the model offered by the 'dominant' discursive ensemble. Although corporate workers resisted the 'learning organization' model and the associated practices (most notably the PIR), there was a sense that the vision of software development this model offered corresponded to what software development was, and was about. There was no question that 'predictability,' 'discipline,' 'measurement,' and '(anti-)bureaucracy' were legitimate discourses. However, corporate workers also recognized (albeit to a lesser extent) that 'learning,' 'creativity,' and 'communication' were legitimate discourses when it came to software development. *Thus, it seems clear that the legitimacy of the discourses of one discourse ensemble was not established at the expense of the discourses of the other.*

There was no struggle at IBTech to either maintain or displace a discursive ensemble in dominance. Rather, the challenge that corporate workers faced was to constitute software development in the most legitimate manner, given the constraints that the Corporation imposed, without denigrating the 'alternative' ensemble.

Ideally, all seven discourses would have dovetailed into each other without contradiction. Software development would have been efficient, yet creative; predicable, yet agile; disciplined, yet able to accommodate changes seamlessly; measured with precision, yet lean; and so on and so forth, ad infinitum. And it was this ideal that corporate workers at IBTech aspired to. Corporate workers hoped that this ideal state had become real through the consumption of models that conveyed these ideas. As a result, a waterfall lifecycle coexisted with a generic iterative lifecycle, and the CMM was found next to an agile development plug-in. The pursuit of this ideal made IBTech a context in which anything capable of communicating commitment to the symbolic universe of the Corporation – its demagogy of efficiency, happy committed workers, infinite flexibility, order, control – was uncritically consumed.

A crucial error that can easily be made is to equate the norms of the Corporation only with considerations of efficiency and calculation-- that is to say, to assume that the Corporation is about "rationalization" in a Weberian sense (Weber, 1968). The popular anti-corporate rhetoric can easily lead us to do so. For example, Ritzer (2000), in 'The McDonaldization of Society,' uses as an analogy the fast-food industry and the success of McDonald's restaurant to explain and criticize the model that large multinational corporations across different industries often follow. In short, the model is the combination of four principles: predictability, control, calculability, efficiency.

To be sure, these four principles are strikingly similar to the discourses which formed the 'dominant' discursive ensemble. It would be tempting to posit that what the Corporation was all about was the pursuit of predictability, control, calculability, and efficiency; or, in the researcher's terms, predictability, discipline, measurement, and (anti-)bureaucracy. It would also be tempting to assume that what fell outside these four principles or discourses was opposed to the tenets of the Corporation. Making such assumptions would be a mistake.

The discourses which formed the alternative discursive ensemble were also part and parcel of the stock of discourse of the Corporation. 'Learning,' 'innovation' and

'communication' are recurring themes in the normative management literature and the business press. To be convinced of this fact, one needs do little more than to open an issue of *Harvard Business Review* or Business Week. Management consultants also restlessly promote techniques and tools intended to make organizations and the individuals they employ learn better, innovate faster, and communicate effectively, regardless of location (Rovik, 2002; Sahlin-Andersson and Engwall, 2002b). Although the main selling point of these tools and techniques remains the conjecture that their adoption will make the organization more effective and efficient at developing software or achieving other instrumental goals, several other themes are also tapped extensively. Consequently, it would be incorrect to consider 'learning,' 'innovation' and 'communication' discourses that went against the code of beliefs of the Corporation and a source of resistance to what the Corporation stood for.

In sum, and contrary to Fairclough, the existence of two objects 'software development' in the study organization does not imply that a 'dominant' discursive ensemble based on the core precepts of the Corporation was challenged by a contending discursive ensemble. Rather, it points to the complexity that corporate workers experienced in reconciling the contradictory ideas that the Corporation promoted. For example, an individual might well have talked up the merits of the CMM/I, and genuinely believed in the model's merits, and a few hours later, have stressed the importance of creativity in software development with the same conviction. Corporate workers held diffuse ideas about what software development was and how it should be practiced.

In recent years, many commentators have contrasted the different ways in which software development can be thought of and practiced by emphasizing extremes. For example, commentators frequently suggest seeing software development on a continuum ranging from agile to plan-driven, or from adaptive to predictive (Boehm and Turner, 2004). By considering some critical factors (e.g., project size and criticality of the product developed), a software organization should be able find and establish an appropriate middle ground. Seen this way, agility and creativity are acquired at the expense of control and predictability.

At IBTech, corporate workers refused to sacrifice agility and creativity for control and predictability. To them all of these qualities were desirable, and their associated discourses were valid within the corporate context. Corporate workers wanted the best of each software development approach. The line of attack they took was to exploit the tensions that existed within discourses in the form of linguistic ambiguity. The linguistic ambiguity that was inherent in texts and discourses enabled corporate workers to go back and forth from one discourse ensemble to another in order to draw on the discourses they needed to formulate acceptable texts.

For example, two different ideas existed in tension within the 'learning' discourse. From a software process improvement perspective, learning occurs when a process is stable, the causes of deviation from prediction are systematically identified, and when the process is improved based in these observations. Seen this way, learning occurs at the process level. On the other hand, learning can be seen as a human phenomenon, facilitated by sound communication among individuals. Learning is here seen as critical for innovation, and as being in tune with the human facet of software development.

The 'predictability' discourse also allowed for a great deal of linguistic ambiguity. A consultant employed by IBTech explained the different meanings the discourses might take in the following way:

> 'Predictable' is at least contestable […] Predictability in terms of plan-driven means that if you draw up a plan which says 'we will analyze, develop, implement' and that says we will deliver in August 6th. We know what we are doing and will know if we are off course. That's one sense of 'predictability' where you are trying to map out the future. Given that you can't have any control over the past, you are trying to exert control over the future. And that's a particular engineering perspective on the world.
>
> [Iterative development] takes it at a totally different level, which say 'try not too much about controlling the big future, think about how you might control the next week, the next month. And work at controlling in small slices.' That way you will achieve the level of control over the future that will mean that you don't get into messes where after eighteen months work you discover that you will not have any chance of making the delivery. [Interview with a technical contractor, 10-12-2004]

Of course, not all discourses possessed the same degree of interpretive malleability. For instance, the 'discipline' discourse unambiguously implied adhering to a process. However, the extent to which the process had to be defined remained flexible. In this sense, all of the seven discourses allowed for some degree of flexibility. Corporate workers could always play with the interpretive malleability of a discourse in order to playfully emphasize some aspects of software development.

7.4 Symbolic power and software development

The notion of symbolic violence is a rather flexible notion which is used in many different ways across Bourdieu's writings. Symbolic violence occurs when the meanings or ways of seeing imposed by a particular group is accepted in a social context. A consequence of the acceptance is that it becomes difficult to develop ways of talking about or engaging with an object that are different from those which have been established. The overall effect of the acceptance, as Bourdieu sees it, is the preproduction of the structure of the distribution of certain forms of capital, and hence the reproduction of the total institutional structure.

Institutions simultaneously facilitate and constrain the discursive practices of their members (Fairclough, 1995: 38). In the case of IBTech, the Corporation provided corporate workers with a mode of expression that enabled relations of communication. More specifically, the Corporation provided particular ways of talking and ways of seeing that were shared by the majority of IBTech's corporate workers. The corporate language provided a standard way of talking about software development and assessing the value of the statements formulated, regardless of the corporate workers' geographical locations or academic backgrounds. However, the Corporation discouraged corporate workers, who were concerned with preserving the capital that was created by the Corporation, from using other languages. As a consequence, the Corporation induced corporate workers to represent software development in a particular way – a way that concurred with its expectations. In this sense, the institution restricted the number of arbitrary ways in which software development could be seen.

Symbolic violence was in operation in the study organization because the use of any modes of expression other than that of the Corporation was discouraged. Other languages could only be used at the risk of facing sanction (i.e. the diminution of the amount of capital accumulated). In providing a predefined repertoire of discourses to represent software development, and in channelling how software development should be talked about, the Corporation structured how corporate workers could understand software development. The Corporation encouraged the adoption of particular beliefs-- for example, the belief that software development has to be made disciplined and that the ability to learn for those involved in software development matters. The discursive constitution of software development, therefore, took place under the influence of a particular social-institutional context and, to a large extent, within the limits set by the Corporation.

Symbolic violence also had effects that spilled over into the practice of software development. As explained in the opening of this chapter, there was a relationship between beliefs and practices. Corporate workers tended to practice software development in accordance with what they believed software development was. For example, at IBTech, corporate workers resisted some of the software process improvement practices (most notably the PIR) because these practices went against their beliefs concerning the degree of bureaucracy that software development should accommodate. Corporate workers occasionally had to enact practices that conflicted, to some extent, with their beliefs; there was, nonetheless, some degree of convergence between the way software development was generally thought of and the way it was practiced. In this sense, it is useful to think of the object as something aspired to, a somewhat ideal representation that orients the choice of practices.

In limiting the manner in which software development could be understood, the Corporation limited the different ways in which it could be practiced. The way in which the RUP platform was received vividly exemplified this point. Corporate workers acknowledged that development work should be carried out iteratively because it ensured (according to the text provided by IBM's sales representatives at least) more disciplined and predictable results. However, corporate workers struggled a great deal to make sense of iterative development, as it seemed to contradict what they believed about software development. In particular, it was not clear to them how

RUP could make software development more predictable if the predictions were worked out throughout a project. Software development was never practiced iteratively in the study organization because it conflicted too much with what software development was understood to be.

The notion of symbolic power is often used to analyze how one class dominates over another by confirming or transforming the manner in which the dominated class perceives and engages with the objects of the social world (e.g. Bourdieu and Passeron, 1977). This is facilitated by the imposition of the language of the dominant class on the dominated class. The case of the discursive constitution of software development at IBTech is interesting because corporate workers formed a uniform class, rather than divided classes seeking to impose meanings that benefitted their interests. The phenomenon witnessed at IBTech represents a case of unification. Unification takes place when a group of individuals is led in practice to accept one specific language as the only legitimate language (Bourdieu, 1991: 44-52). This language becomes the norm against which all discursive practices are assessed. It involves the operation of an institution powerful enough to impose the universal acceptance of the legitimate language (Bourdieu, 1991: 46). In the present case, the stranglehold of the Corporation restricted the possibility of using languages other than the corporate language to produce and interpret texts about software development.

In sum, the constitution of software development in keeping with the tenets of the Corporation represented an act of symbolic violence. Symbolic violence, which involves the complicity of the dominated, was in evidence in the acceptance of the corporate language as the sole language legitimate to talk about software development. The uncritical acceptance of this language had the effect of directing attention to specific discourses and constrained the manner in which corporate workers engaged in the activity of software development.

8 Conclusion

Software development is today thought about and executed in a variety of ways under such names as agile, open source, and plan-driven. There is, however, surprisingly little empirical knowledge about how and why people come to think about software development in a certain way and adopt certain practices. Practitioners certainly tend to embrace the software development approach that they believe most suitable, but how this perceived suitability is collectively formed in real organizational contexts remains largely misunderstood. This issue, it is argued, has far-reaching implications as it touches the core of what software professionals across functions and industries do. The purpose of this research was to elucidate how actors' understanding of software development and the legitimacy of the practices they adhere to is shaped.

The concluding chapter provides an overview of the thesis as a whole. It also outlines the core contributions that this research makes to the literature and the practice. Finally, the limitations of the research are discussed and some key areas for future research are identified.

8.1 Overview of the dissertation

To assist the reader, this section restates the research question and summarizes the major points developed within the preceding chapters. The dissertation began with an introduction to the rationale supporting the formalization of software development practices. In essence, it was the large number of project failures that provided an impetus for the development of systematic development practices. Since their inception, however, there has been a sentiment that systematic development practices impose unwanted restrictions on the software development process. The question of how to best practice software development is still today highly topical.

The literature review in Chapter 2 showed us that software development has been thought of and practiced in very different ways over the past four decades. A chronological review of the most influential ideas and texts revealed that the methods

and applications of process and quality management have played a significant role in defining how software development should be practiced in different contexts. The academic literature corroborated this observation and suggested that beliefs inform the adoption of development practices. However, it is not clearly understood how beliefs about software development and software development practices come to be established as legitimate in an organization. In the backdrop of this topic, the research question informing the study is concerned with the establishment of beliefs about software development and software development practices within an organization.

The theoretical framework of the study was established in Chapter 3. Organizational discourse theory was identified as being highly appropriate to examining how the legitimacy of beliefs and practices are negotiated. Critical discourse analysis and the theory of symbolic violence were introduced and provided the analytical constructs needed for this research. The theoretical framework was designed with the goal of remaining analytically sensitive to the operation of power in language use and revealing the interaction between (discursive) practices and the wider socio-institutional contexts in which they occur.

Chapter 4 presented the interpretive research methodology and the case research strategy. A qualitative analysis of the texts that organizational actors produced, transmitted, and interpreted as part of the activity of 'software development' was adopted and complemented by observational data. Ricoeur's hermeneutics of suspicion was selected as a paradigm of text interpretation. This paradigm enriched the analysis by encouraging the researcher to develop plural plausible readings of the organizational texts and explore the interpretive flexibility of discourses inhabiting them.

Chapter 5 presented the study organization and provided an overview of the evolution of the 'learning organization' program. This program was intended to standardize software development practices across IBTech's ten software houses. In a first stage, following the precepts of the CMM/I, IBTech attempted to build its standard software development process from internal best practices. The organization later went on to acquire two commercial methodologies. Throughout Chapter 5,

particular attention was paid to the organization's software process improvement practices in order to provide a concrete illustration of the difficulties of creating and adapting a standard software development process.

Chapter 6 outlined the seven organizational discourses inhabiting the organizational texts collected and analyzed. The chapter also highlighted the tensions and connections that existed among discourses, as well as the contradictions that existed within discourses. An analysis of the manner in which the discourses relate to one another revealed that they constitute two conflicting objects 'software development' – one 'dominant,' the other 'alternative.' Participant observation data provided the researcher with the holistic understanding of the socio-institutional context needed to theorize the findings.

The case analysis in Chapter 7 revealed that a struggle for professional recognition waged within IBTech led actors to adopt normatively-sanctioned practices. Actors were successful in this professional struggle to the extent that they possessed the ability to understand what was expected of them in terms of behaviour and to communicate this understanding through their practices (including their discursive practices). As such, software development acted as a platform for professional recognition. Bourdieu's concepts of 'field of struggles,' 'field of forces,' and 'capital' were employed to illuminate the relationship between, respectively, the professional struggle in which actors were engaged, the logic of the broader socio-institutional context, and the competences that actors possessed.

The analysis showed that individuals had absorbed a system of norms that regulated their discursive practices. It was argued that the constitution of software development, in keeping with the logic of the corporate language, represented a form of symbolic violence. The uncritical acceptance of this language had the effect of directing attention to specific discourses, and constrained the manner in which people could execute software development.

The case study showed that it is through a discursive process that the actors' understanding of software development and the legitimacy of the practices they adhere to is shaped. This process involves balancing creativity and flexibility with

efficiency and predictability. In the present case, however, the use of the legitimate language encouraged the simultaneous adoption of the two conflicting visions of software development.

In the next section of the chapter, we consider the implications of the study for research and for practice. We start off with a retrospective assessment of a discourse analytical approach that is grounded in the day-to-day practices of organizational actors. Then, the merits of a discourse theoretical approach informed by Fairclough and Bourdieu's ideas for research are discussed. On a more practical note, it is argued that the insights that the study produced have significant implications for practitioners seeking to transfer knowledge within organizations, and to balance agility and discipline agility and discipline in software development.

8.2 Contribution of the research

The research contributes to the knowledge base of the information systems discipline in three principal ways: (1) by providing rich and contextually-grounded insights illuminating how beliefs about software development and software development practices come to be established as legitimate, (2) by proposing a discourse theoretical approach, and (3) by highlighting how the transfer of internal knowledge can be facilitated in a geographically-distributed organization and how a compromise between agility and discipline can be found in a software organization. These are, respectively, the methodological, theoretical, and practical contributions of the research.

8.2.1 Methodological contribution

Information system development has remained one the most widely-researched areas in the IS field (Avison and Fitzgerald, 2006). Because of the close link between the research and the practice in this area, IS researchers have traditionally been preoccupied with finding the practices (including methods and tools) that provide superior technical results for certain types of projects, without paying much attention to social and organizational factors that define the perceived appropriateness of such practices in their context of use.

Even in rare cases where social and organizational factors are taken into consideration when examining the engagement of actors in the activity of software development, it is more often than not an examination that is positivistically motivated, as several authors have noted (Nandhakumar and Avison, 1999; Russo and Stolterman, 2000b). Typically, the studies' objective is to provide causal explanations between variables (e.g., Iivary and Huisman, 2007; Serour and Henderson-Sellers, 2002). Usually survey-based, such studies are characteristically conducted across large populations of organizations. Furthermore, several of these studies are normative in orientation, and seek to facilitate the adoption of methods and the implementation of tools deemed to provide efficiency gains.

In spite of a marked interest in what software professionals do and should do, the IS literature says almost nothing about the social process through which these actors come to regard particular development practices as being appropriate. All in all, rich descriptions of the behavior of software professionals are lacking (Dubé and Robey, 1999; Gasson, 1999; Madsen et al., 2006; Nandhakumar and Jones, 1997; Russo and Stolterman, 2000a). Few commentators recognize or address the critically-important social, political, and organizational dimensions of software development. It is the researcher's contention that this caveat in the literature is, in part, due to the fact that IS research has traditionally been dominated by a positivist orientation, and that questions such as those addressed herein, though highly relevant for the theory and practice of software development, can hardly be answered by using the methods of the natural sciences. Thus, in order to contribute significantly to the theory of software development, it was necessary to transcend the prevailing positivist orientation and develop a rich and contextually-grounded understanding of the practices of actors involved in software development.

The interpretive approach emerged as a necessary choice for developing a finer understanding of the phenomenon. Retrospectively, looking at practices at a distance from the context could not have told a complete story. However, asking actors about their beliefs and intentions in the context of an interview could not have produced unsatisfactory insights either. The problem, as countless scholars have observed, is that informants are sometimes totally unaware of certain aspects underlying many of their own activities, and have little real understanding of the phenomenon that

scientists are interested in (Denzin and Lincoln, 1994; Van Maanen, 1979). The difficulty is attributed to the taken-for-granted nature of the social world for those who are situated in it.

The question of how to produce knowledge of the social world which is not reducible to the practical knowledge possessed by lay actors, or which breaks with the immediate experience of the social world, has generated a vigorous debate among social scientists. It was argued in the methodology chapter that one way to avoid the main limitations of the two dominant modes of knowledge production (what Bourdieu calls 'subjectivist' and 'objectivist') is to focus on the practices of actors in the context in which they naturally occur. For Bourdieu, adopting a practice perspective involves taking into consideration, on the one hand, the interactions of actors and the structures of the field, and, on the other hand, the actors' dispositions to produce particular practices and their perceptions that result from their inhabiting the field. The research operationalized this practice perspective.

So, rather than focusing on the "provoked narrative" of actors, this interpretive research concentrated mainly on their actions – their practices in situ (Czarniawska, 1992). More precisely, it concentrated on the actors' discursive practices and the traces they leave (i.e. the texts) (Ricoeur, 1991). The choice of a discursive approach was particularly appropriate because interactions are principally mediated through language in professional service firms such as IBTech. Thus, in order to develop a context-based interpretation of the manner in which beliefs are articulated and tied into the activity 'software development,' the focus was placed on the texts that actors produced, circulated through more-or-less formal channels, and interpreted. These texts were generally about software development since they were collected in the context of software development. In this sense, they reflected and were indicative of the prevailing beliefs and of the development practices adopted at a particular point in time.

If one seeks to understand the intricate manner in which beliefs and practices come to be socially established as legitimate, one should ideally seek prolonged exposure to the context in which this process unfolds or has unfolded. The researcher had the unique opportunity to have not only exposure to the social-organizational context in

which software development took place, but also a first-and view of the SPI process – the very process designed to establish the legitimacy of software development practices at IBTech. The merits of this line of inquiry are well-established in many disciplines, and are seen increasingly in IS research. However, as far as our understanding of software development is concerned, evidence for the merits of this line of enquiry still needs to be presented (Nandhakumar and Jones, 1997; Ronkko et al., 2002). In presenting this evidence, this dissertation contributes to the knowledge base of information systems research.

Another contribution was made in the area of data interpretation. The interpretation of organizational texts was conducted under the assumption that a text may take several different meanings. As such, it was deemed futile to attempt to extract the 'true' meaning of the texts, which is usually assumed to be what the text producer meant (Gadamer, 1977; Habermas, 1980). Ricoeur's hermeneutics of suspicion proved very useful in making sense of the texts and letting discourses emerge from them. Adopting this paradigm of text interpretation invited the researcher to recognize his ability to act as objective interpreter and encouraged him to develop plural readings of the same texts. The texts were revisited several times through hermeneutic circles in order to go beyond what appeared immediately obvious in them.

Adopting Ricoeur's hermeneutics enabled the researcher to understand better the texts and the interpretive flexibility of the discourses inhabiting them. It is by employing this mode of analysis that the researcher became aware that a discourse could carry different meanings. In particular, it is through a Ricoeurian engagement with the texts that the researcher discovered how, in this particular organization, it had been possible to espouse two different visions of software development, rather than to sacrifice efficiency and control for agility and creativity.

Although hermeneutics is not unknown to the field of IS research (Boland, 1991; Lee, 1994; Myers, 1995), the application of the hermeneutic of suspicion represents an addition to IS research because the principle of suspicion is by far the least-developed in the IS research literature (Klein and Myers, 1999). The dissertation

demonstrated the application of a potent means of engaging with organizational texts that was little known to the field of IS research.

8.2.2 Theoretical contribution

Adopting a discursive lens, this dissertation has posited that software development has an important linguistic dimension. From this perspective, the mundane production, distribution, and consumption of texts by members of the study organization was part and parcel of the activity 'software development.' The dissertation has demonstrated that an object 'software development' was shaped as the result of the discursive practices of individuals. The discursive lens offers a novel perspective for looking at the process through which certain ways of thinking of and practicing software development are constituted. This lens is of theoretical value because it helps us to shed light on the long-standing open question of how actors come to adopt certain beliefs about software development and how certain software development practices come to be established as legitimate. Developing the right framework was critical to developing a meaningful and theoretically-rigorous interpretation of this complex human process.

There exists a vast array of different discourse analytic approaches. These approaches differ in the manner in which they conceptualize agency and to the extent to which they focus directly on the dynamics of power. Furthermore, approaches differ to the extent to which the broader context is deemed relevant to the analysis of the texts (Hardy, 2004; Hardy et al., 2005). The theoretical framework adopted is distinctive in that it rejects barriers between the micro discursive events and the macro structures. A key analytical contribution the framework enables is to effectively relate the wider institutional context with the interests of organizational actors. In fact, one of the key objectives actively pursued by the researcher was to better understand the constitution of software development by overcoming the simplistic opposition between agency and structure. By drawing on Fairclough and Bourdieu, and by combining the ideas of both authors in such a way that their respective strengths offset their limitations, the study showed that individuals are motivated and torn from a state of indifference by the stimuli sent by a certain field – and not others. At IBTech, the great desire to improve one's status, and the need to

develop a positive professional image in order to do so, led individuals to enact particular discursive practices. The statements formulated by actors, and the manner in which these statements were formulated, represented practices to be appreciated--signs of possession of valued capital. Thus, the study showed that the context matters and that it cannot be analytically separated from the agency of organizational actors.

The study conjectured that individuals neither acted of their own free will, nor that their actions were mechanically instantiated in response to social structures. To articulate this idea, it was argued that it is appropriate to think of the relation between the actor and his social world as one of mutual possession. The concept of 'corporate worker' was instrumental in articulating this idea clearly. IBTech's organizational actors were not ordinary IT professionals; rather, they were corporate avatar. They were, in essence, self-interested creatures caught in a dynamic within which being good to the Corporation equalled being good to themselves. Software development provided an occasion to embellish their own professional image (through a display of valued competences) and positively distinguish themselves from others.

All in all, the theoretical framework proved particularly efficient at unpacking the manner in which the context – or, in analytical terms, how the actors' investment in a field – came to shape the actors' understanding of software development and the legitimacy of the practices they adhered to. The case analysis showed that the context might not have to be thought of as operating autonomously and externally from the agents, or as something that exerts its effect without the willing participation of purposeful individuals. From the discursive perspective adopted, agents and their discursive practices were embedded in the context.

In order to relate the context to the discursive practices in as rich a manner as possible, several detours were necessary. Drawing on the deep understanding acquired through an in-depth immersion in the field, the researcher expounded the prevailing organizational values. More specifically, it was explained what working for a leading investment bank meant to the corporate workers and the pride they derived from it. It was also described how an ideal of professionalism and a meritocratic ideology were consumed. With hindsight, the theoretical framework proved to be very effective in bringing these ideas together, and enabled the

contextualization of the discursive practices. For the point was not only to appreciate what the actors said, but to understand why they said what they said.

The theoretical framework also allowed for important nuances. For example, and contrary to common wisdom, it was clear that hierarchical ranks did not determine social positions. Hierarchical ranks were the result, but not the cause of, social positions. The notion of capital proved particularly useful in this respect. The analysis showed that hierarchical ranks were the result of the accumulation of corporate capital, which was also required for development of a viable professional image. Whatever their position in the formal hierarchy, corporate workers derived gratification from demonstrating their ability to produce discourses that are valued. It flattered their egos and made them feel professional, competent, important, and potent. The dynamic is complex, yet fundamental to understanding the discursive constitution of software development.

In summary, the value of the theoretical framework rests in the incisiveness and richness of the interpretation it enabled. A discursive approach informed by the work of Bourdieu and Fairclough permitted a detailed and sophisticated consideration of issues of power and of the relationship between the micro and the macro. To date, the literature on software development has not incorporated such a perspective in spite of the advantages it presents for better understanding the establishment of particular practices. By demonstrating the value of a discursive approach informed by the work of Bourdieu and Fairclough, a theoretical contribution was made (Barrett and Walsham, 2004).

8.2.3 Practical Contributions

The internal transfer of knowledge and the standardization of practices

Programs intended to enable the internal transfer of knowledge (i.e. good practices), such as those undertaken by IBTech, have been widely adopted by software organizations. These programs typically aim to transfer knowledge among several geographically-distributed units or, increasingly, from a parent organization to offshore development teams in outsourcing context.

As the case of IBTech illustrates, however, the internal transfer knowledge poses several challenges to practitioners. Knowledge for software development often has a tacit component. Consequently, knowledge may be rooted in experiences and idiosyncratic personal relationships and be difficult to communicate to the rest of an organization. Even in cases where the source unit is willing to share knowledge, the recipient units may not be willing to discard old practices and sustain new ones (Szulanski, 2000). This situation was particularly evident at IBTech. The case of IBTech also shows that identifying and communicating knowledge and good practices, even in a case where formal channels exist, may be a burden for software professionals. It may distract software professionals from what they consider their normal activities, especially when producing additional documents and attending meetings is required.

In the mainstream of management literature, the difficulties have principally been associated with difference of language, cultural conventions, and identities. The solution is typically formulated in terms of sound leadership. The argument is put that if managers could nurture a cohesive set of sociocultural practices, then several of the difficulties associated with transferring knowledge would be overcome. This view is widely-accepted in the strategic management literature. For example, Ghoshal & Barlett (1988) argue in a seminal article that "normative integration" between different parts of a firm is key for the effective diffusion of innovations (including processes) within an organization:

> High levels of normative integration and information exchange can enhance the salience of the convergent interests and […] lead to more vigorous participation of the subsidiary in the tasks of creating, adopting, and diffusing innovations that benefit the company as a whole. In the absence of such integration, however, the conflicting interests may become relatively more salient […] (Ghoshal and Barlett, 1988: 386)

Similarly, Szulanski (1996) observes that the transfer of best practices inside the firm is facilitated by homogeneity among individuals. Common norms and the use of a shared language make relationships straightforward and simulate the exchange of knowledge:

> The success of [individual] exchanges depends on some extent on the ease of communication and on the 'intimacy' of the overall relationship between the source unit and the recipient units. (Szulanski, 1996: 32) […] shared meanings and behaviors facilitate coordination of the activities, making behaviors understandable, predictable and stable. In this way, new practices become institutionalized. (Szulanski, 1996: 29)

More recently and directly apropos of software development, Levina & Vaast (2008) observe that it is accepted in the literature that the differences in identities and language create impediments for effective collaboration among teams spanning multiple geographies (cf. Levina and Vaast, 2005; Orlikowski, 2002). Thus, software organizations wanting to transfer knowledge should strive to cultivate a common mode of expression and shared norms among their members.

Significantly, many of the factors deemed capable of enabling the efficient transfer of knowledge were present at IBTech, including strong normative integration, shared identities and the acceptance of the corporate language to talk about software development. Yet, transferring knowledge and practices so as to constitute a standard software development process proved unfeasible.

A wider implication of these findings for the practice is the proposition that standardizing software development goes beyond merely standardizing software development practices. It also involves standardizing the beliefs that software professionals have about their work and ensuring that these beliefs are congruous. The case demonstrated that in spite of the prevalence of a common language, shared identity and cohesive norms, individuals (and even an individual) may hold diffuse ideas about what software development is.

Leaders must be attentive to the actors' competing and divergent visions of software development, all of which may very well be legitimate in the context in which their organizations operate. Understanding the different visions of software development may be achieved by paying attention to the manner in which software development is depicted by software professionals, and in the texts they are constantly exposed to. This task becomes salient as organizations increasingly conduct several initiatives in parallel and pursue mixed objectives, such as increasing process maturity and

becoming more agile. While following contradictory objectives may appear to be symptomatic of a lack of focus, it reflects the reality of many software organizations. In practice, software organizations not only have to develop products efficiently, but also have to meet the normative expectations of their field (Adler, 2006; Meyer and Rowan, 1977).

For software organizations seeking to standardize its practices, the starting point should be to establish a vision of the development approach to institutionalize. The vision must provide a strong focus on how the practices are to contribute to the development approach aimed at. This way, instead of attempting to transfer unrelated software development practices, the firm will concentrate on preserving and transferring the practices that contribute to getting closer and, ultimately, to attaining, the vision of software development aimed for. In proceeding this way, the practices will be judged by the firm according to how well they contribute to getting closer to the vision, rather than by only how effective they are in a particular context.

This vision of software development was missing at IBTech. Good practices were selected according to whether they provided good results in a particular context; however, the local good practices that populated the 'SEPG CD system' never formed a coherent whole. Consequently, after two years of sustained effort and major investments, no standard process had developed.

Working around a vision also provides a sense of direction as to how to proceed to preserve and transfer practices. For example, if attaining a high level of process maturity is commensurate with the vision, then a traditional SPI approach like the one used at IBTech is likely to be appropriate. However, if the vision leans towards developing the ability to respond quickly to change, then more informal practices, such as peer programming and the daily sunrise meeting, would probably be more appropriate to foster the transfer of knowledge between individuals. The point is to avoid relying on practices for the transfer of knowledge that clash with the vision aimed for, and that are likely to give rise to resistance.

In summary, the message this dissertation conveys with regard to the transfer of knowledge and the standardization of practices is that actions should be guided by a

vision of software development. Working from a vision helps to ensure that the practices transferred are coherent and can lead to the development of a practical standard software process. It is true that organizations may have to pursue more or less inconsistent initiatives simultaneously. This reality should not so much be seen as an impediment to the establishment of a vision, but rather as a reality that confirms the crucial importance of proceeding with a clear vision. One of the tasks of leaders is to communicate the vision across sites and monitor conformance to it by looking at how software professionals discursively constitute software development. Leaders must also ensure that the mechanisms for the transfer of knowledge are congruous with the vision.

Balancing agility and discipline

Software professionals assuming a managerial role are confronted with many challenges. First, managers are under pressure to develop their organization's ability to deliver software better, faster, and at less cost. At the same time, they face a demand for increasingly sophisticated products. This is also true for organizations, such as JP Morgan Chase, which are not software companies, but which find that much of their operations rely on software. These challenges are further complicated by the fact that organizations often span geographically, thus creating the need to acquire the means to enable coordination between software houses.

The pressure to become more efficient and the increasing complexity of the products, coupled with the need to enable coordination between individuals, provide a justification for adopting disciplined approaches to developing software (Parnas and Clements, 1986). Methodologies are still today the tools of choice in the attempt to more systemically organize the design and construction of software. However, practitioners have expressed the concern that the discipline recommended by methodologies is sometimes experienced as burdensome and coercive constraints. Critics have argued that adherence to the methodology rule book stifles the motivation and creativity that are, over the long run, required for high-quality software development (Introna, 1996; Wastell, 1996).

In the literature intended for software practitioners, there is increasing interest in the question of how to balance discipline and agility. This interest stems from the

realization that addressing the challenges that software development poses is not about making the activity as disciplined as can be imagined. For managers, methodologies have historically played-- and perhaps, now more than ever, play-- a central role in finding a practical middle ground between control and flexibility.

That some software development methodologies are appropriate in some situations, but not in others, has the earmarks of a truism (Avison and Fitzgerald, 2006). In this light, the problem for practitioners is generally seen as being about determining which methodology to use given their own situation. In order to help in this practical decision, researchers have classified methodologies according to the situations in which they are most appropriate (Avison and Taylor, 1997). The selection of a suitable methodology should thus involve a fairly rational choice process whereby particular projects and organizational characteristics determine an acceptable choice. And this is how practitioners are, in the main, expected to balance discipline and agility.

The recent work of Boehm and Turner (2004) offers a case in point. The authors argue that there are five "critical factors" involved in determining the relative suitability of agile and disciplined methodologies. These are project's size, criticality, dynamism, personnel (i.e. skills), and culture factors. Boehm and Turner (2004) consider the critical factors to be unbiased by commercial interests and urge practitioners to use them to find a sensible balance between agility and discipline. Hence, practitioners are presumed to select methodologies relatively rationally, in the sense of having their selection process guided primarily by an efficiency concern.

However, this view fails to grasp the significance of the political behavior of software professionals. It depoliticizes the process by which balancing agility and discipline unfolds. In doing so, it obscures the fact that exploring the space between alternative visions of software development can be linked to personal interests. Denying the political dimension of selecting a methodology promotes a limited understanding of the question and prevents software professionals from assuming a management role to act effectively. This dissertation, in contrast, draws attention to the political nature of the process by which agility and discipline are negotiated and the role methodologies play in this process.

As the case analysis showed, personal interests fundamentally motivated corporate workers. Corporation workers were primarily concerned with constructing a positive professional image and elevating their status. These objectives were the stakes that the corporate field offered, and pursuing them involved the use of some form of power (i.e. capital). In this sense, the pursuit of personal interests and the use of capital permeated the constitution of the object 'software development' and software development practice. The balance between agility and discipline accidentally emanated from this struggle.

Once dismissed as counterproductive, it has been recognized that organizational politics are inevitable and not necessarily a bad thing (Mintzberg, 1984; Pfeffer, 1981). In the normative management literature, it is a frequently repeated statement that the challenge for leaders is to align the interests of actors with those of their organization in a 'productive way.' In the present case, the interests of corporate workers were almost perfectly aligned with those of the Corporation. As a result, a rather disciplined form of software was maintained in dominance. However, whether relying on a relatively disciplined approach was a good thing, and whether the organization would have benefited from more flexible practice, remains an open question.

The case of IBTech is illustrative of the reaction of software professionals to the introduction of methodologies intended to provide the benefit of common organizational practices. Important lessons for the practice can be derived from the case. First, it is interesting to note how actors responded to RUP and the agile module. The principles of these two methodologies were at odds with many principles that the 'learning organization' program promoted, which created a dilemma for corporate workers to resolve. On the one hand, they could ignore or challenge the methodologies, but only at the cost of losing capital. On the other hand, they could embrace the principles of these methodologies, yet without turning their back on other ideas and practices they knew to be legitimate. The latter approach proved to be more diplomatic and politically astute and did not involve the reduction of capital.

In the same vein, the case study suggests that balancing discipline and agility may very well be an ongoing process. It seems hard to believe that all members of an organization like IBTech could agree that the methodology they used provided a *juste milieu* between discipline and agility. Although the idea of a perfect compromise is seductive, whether it can be attained and maintained is questionable. In the study organization, the need to change some practices was constantly being stimulated as ideas circulating in the environment (agile and iterative development) received interest, created dissatisfaction with the current state, and encouraged experimentation. Some corporate workers sought to maintain a status quo; others attempted to change it. In this process, the agile-discipline balance was revisited. This dynamic was particularly evident when the agile consultant employed by IBTech outraged the actors responsible for the 'learning organization' program by criticizing the CMM/I.

The case analysis points to the imagination of organizational actors in finding a practical solution to the intricate question of how software development should be thought of and practiced. Leaders must be aware that status and career prospects may be attached to the question and that it may have far-reaching implications for individuals. Leaders must, therefore, be wary of the claim that methodologies are instruments capable of providing the desired balance between agility and discipline. In reality, as the case of IBTech vividly demonstrated, the balance is not embedded in the methodologies as such; rather, it is negotiated as individuals juxtapose their interests with the discourses that the methodologies convey.

8.3 Limitations

As is the case with any research, this study has several limitations. What, exactly, these limitations could be seen as being might depend, to a large extent, on the reader, and on his/her scholarly interpretation of what good research is, and what criteria should be used to assess the quality of the research. Addressing these limitations provides the researcher with an opportunity to clearly state his position on them.

8.3.1 Question of generalizability

For many scholars, the key limitation of studies based on a single case study is that the findings do not generalize. The basis for this judgement is that there is no ground for assuming that what was observed or discovered in the study's setting is applicable to other organizations (Whetten, 1989). From such a standpoint, the conclusion reached in this doctoral study remains highly specific to the setting in which the study was conducted. And since the objective of any scientific endeavour is to generate knowledge that can be used as a basis to understand a phenomenon in settings different from the one where it was confirmed, the present study is of little scientific value.

This line of reasoning implies a particular model of science (Orlikowski and Baroudi, 1991) which has not been adopted for this doctoral research. Rather than seeking to produce abstract generalizations about the behaviour of people involved in software development, the present research sought to generate a justified interpretation of a phenomenon in order to improve the appreciation of a community of researchers. Here, the key determinant of the value of a research is whether the interpretation suggested is well-reasoned and can be justified to a knowledgeable cynical audience (Barrett and Walsham, 2004; Klein and Myers, 1999). The knowledge that is developed through the research process may or may not hold true to other organizations, but this is not the relevant question (Walsham, 1995). The relevant question is, rather, whether the claims made by the researcher based on the data collected and presented are plausible and whether they are of interest to an academic community (Lee, 1999).

Although the study was not conducted in order to produce abstract generalizations about the behavior of people involved in software development, it is believed that the knowledge the study produced may be useful to make sense of what happens in other organizations (Lee, 1999; Lee and Baskerville, 2003). The study may be useful not in the sense of explaining in mechanistic terms, but in the sense of providing richer insights into a socio-organizational phenomenon and demonstrating the value of an analytical approach. It is the researcher's contention that the study has contributed to developing awareness of some dynamics needed to better understand the standardization of practices in other knowledge-intensive organizations.

8.3.2 The concepts of corporation and corporate field

Strictly speaking, a corporation does not have to be a capitalist enterprise. Nor does it have to be privately owned. Churches, municipalities, and universities can be corporations. The term 'corporation' was used in this dissertation as it is used in common parlance to refer to a large business corporation, and implied a form of business enterprise. It is recognized that the concept of 'corporation' is complex and that the term can mean different things to different people. For example, it can be a way of organizing, a legal entity, and a means to centralize resources in the hands of a class (McCraw, 1997; Roy, 1997).

The studies would have benefited from an in-depth theorizing of what the corporation as a social institution (the Corporation with a capital 'C') implied. In this institution, a distinctive logic prevailed. In addition to an obvious concern for efficiency and profitability, the Corporation implied a set of norms and values and a particular way of seeing and engaging in the social world. It implied a way of life. IBTech and its parent organization were posited to be archetypes of this social institution. The life of IBTech's members was assumed to be typical of those working for other corporations. Future research might examine in greater detail the features of the corporation as a normative institution.

Had time and space allowed, the research would have benefited from an empirical demonstration of the existence of the Corporation and of the corporate field. This demonstration might have been conducted according to the procedure that Bourdieu advocated.

Bourdieu states that the boundaries of a field should be determined by an empirical investigation. Such an investigation entails identifying the interests that are distinct to the field. With regard to the corporate field specifically, it should entail looking at the factors that lead young professionals to complete an internship in corporations, study for an MBA, or enter the corporate field, rather than another. It should also involve identifying the forms of capital that are valid in the field and the point where their value evaporates. Consequently, the researcher should provide evidence of what determines statuses within the field. Finally, common distinctive traits should be found among participants, e.g. similar views, tastes, practices (Bourdieu, 1984).

Common traits may be found by looking at consumption practices, professional values, and language.

The researcher did not conduct an empirical analysis of the Corporation. Rather, the corporate field was constituted by the researcher through his experience at IBTech and background knowledge of the 'corporate world' (i.e. the world of knowledge work done for a large multidivisional business corporation). An empirical investigation like that advocated by Bourdieu would have provided more concrete evidence of distinctive practices, including the use of language in the corporate context, and would have added weight to the argument presented herein.

8.3.3 The relationship between beliefs and practices

The doctoral dissertation assumes that there is some correspondence between beliefs and practices. This assumption penetrated the study from two fronts. First, it was acquired from the literature review presented in Chapter 2. The historical review showed that software development practices and discourses have coevolved. The second part of the literature review on the formation of belief and practices demonstrated that the IS literature recognizes that beliefs influence the adoption of development practices. This idea was particularly prominent in Hirschheim et al.'s (1996) article. The authors submitted that the beliefs that actors bring to the development activity legitimize the use of certain tools and methods, and, therefore, certain practices.

Secondly, the assumption was acquired from the theoretical framework. Fairclough contends that it is when the dominance of an ideological-discursive formation (see Table 2 in Chapter 3) is unchallenged that the practices it presupposes become most naturalized. Similarly, Bourdieu's notion of symbolic violence implies that the meanings attributed to objects structure the manner in which actors engage with them and discursively represent them.

The present case would have been more interesting if the object 'software development' had evolved during the course of the study and if the associated changes in beliefs had triggered noticeable changes in development practices. The

case is somewhat static. Consequently, it does not explain the process of change as such, but provides a post hoc explanation for an observable state. The case analysis offers an interpretation of why development practices did not change significantly in spite of an attempt to standardize them.

Although the objective of the study was not to explain the change process, but rather to elucidate the establishment of beliefs and practices, a case showing how the establishment of beliefs and practices unfolded would have been more stimulating. The chances of coming across more dynamic episodes would have certainly been higher if several case studies had been conducted. On the other hand, conducting many case studies could not have been practically done within an acceptable timeframe without forfeiting some of the richness of the data.

8.4 Future research: Beyond software

The work conducted for this doctoral research will develop into a broader research program on innovation development. The program, as the researcher sees it, will focus on the socio-political and context specificity dimensions of innovation development. The program's aim will be to advance theoretical and methodological development by demonstrating how Bourdieu's ideas and a discursive lens can significantly contribute to the development of more sophisticated and detailed interpretations of the innovation related-issues in economics organizations (including software organizations).

Recent developments in the innovation literature increasingly attempt to understand the intricate relationship between innovation development practices and the context in which they are used. A theoretical approach that is highly commensurate with that utilized for the present study posits that innovation depends on the knowledge that is embedded in (discursive) interactions and situated practices. From this perspective, the ability to innovate depends crucially on the ability to share and integrate knowledge within and across organizations. In the coming years, this doctoral research will be used to advance theoretical development in this emerging area, which is known as the "knowledge-based perspective" or the "interactive perspective" on innovation (Newell et al., 2006; Swan and Newell, 2000).

As demonstrated in this study, the 'field' is a powerful notion for conceptualizing the context. The field is constituted, in part, by the adoption of common practices. These practices determine who are part of a field and who are outsiders. As such, several fields may exist within an organization, making it challenging to exchange the knowledge that practices embed. The value of the notion of field is increasingly being recognized within organization studies and is extremely promising to bring light on knowledge-related issues in innovative organizations. Future research might want to look at the characteristics of the field(s) to better understand the micro-processes of knowledge sharing and integration within and across innovative organizations.

A dimension of the notion of 'field' that organization theorists have neglected is the struggle it implies. This notion is generally used to delineate a context in which cohesion exists and to highlight the difficulty of transferring some form of expertise across different contexts (e.g., Levina and Vaast, 2005; 2008). Thus, frictions occur only across fields, not within. This dissertation provides evidence that there is a need to account for the struggles that may occur within fields. In a related vein, the knowledge-based perspective on innovation outlined above says little about power relations. Individuals within a community sharing a similar form of knowledge are generally presumed to have common interests. Moreover, these interests are generally presumed to relate directly to a legitimate organizational goal (Contu and Willmott, 2003; Fox, 2000). This dissertation provides evidence that there is a need to re-examine this assumption through in-depth description and analysis of the practices of actors.

The notion of situated knowledge – knowledge as being embedded in interactions and situated in the practices of actors – has its roots in "situated learning theory" (Brown and Duguid, 1991; Lave and Wenger, 1991). Situated learning theory encourages a focus on the enculturation process and the practical embeddedness of knowledge. However, in the manner in which Lave and Wenger's idea have been popularized, relations of power are dimly recognized or discarded (Contu and Willmott, 2003; Fox, 2000). From a Bourdieuan perspective, knowledge is both situated and necessarily implicated in power relations-- that is to say, in the relations

between individuals possessing unequal volume of capital. More specifically to innovation development, Bourdieu's ideas have the potential to enable the development of a more sophisticated account of the effect of power while recognizing the embeddedness of knowledge in practices. This dissertation has offered a preview of how power might be linked to the themes of innovation and knowledge. Future research might pursue in this direction.

There is a need to better understand how the characteristics of the context may act as barrier to knowledge transfer. Organization theorists have shown that boundaries of fields created by discontinuities in interest and practice are impediments to the exchange of knowledge (Bechky, 2003; Carlile, 2002; 2004). This idea has been useful to shedding light on the challenges of transferring knowledge and collaborating in multiparty information system development projects (Levina and Vaast, 2008). However, the case analysis suggests that discontinuities in interest and practices do not tell the whole story. The study shows that the norms of the field itself may be a key impediment to effective knowledge transfer (see also Metiu, 2006). Despite the prevalence of fairly uniform interests and practices, the corporate context encouraged distant and formal relationships, rather than intimate and authentic relationships which are believed to facilitate knowledge transfer among individuals (Lave and Wenger, 1991). These observations are of crucial importance for scholars seeking to understand through practice theory-based framework how impediment to knowledge transfer may be overcome. Future research might investigate how characteristics of the field, rather than differences between fields, facilitate and hinder knowledge transfer in software development (see for example Szulanski, 1996).

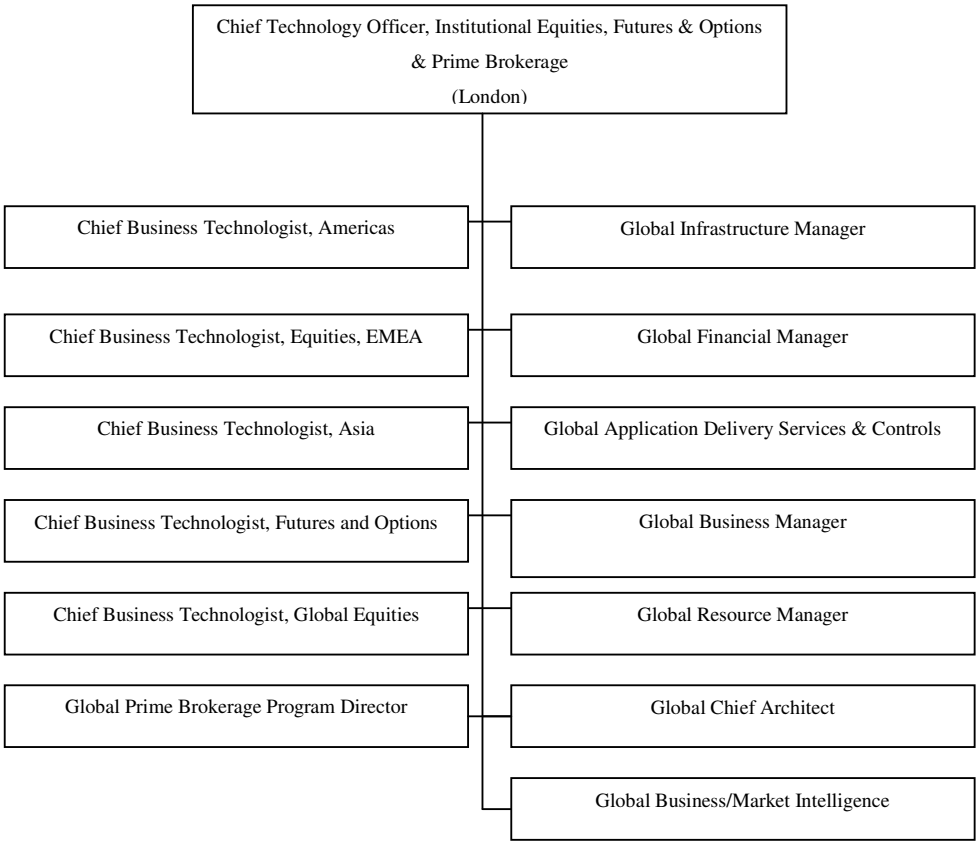
Central to the knowledge-based perspective on innovation is the idea that developing more interactive and collaborative modes of working facilitate knowledge diffusion (Newell et al., 2006; Swan and Newell, 2000). Future research might investigate how the use of Web 2.0 technologies and other such collaborative technologies facilitate (or impede) knowledge transfer and innovation. In the same vein, there is also a need to better understand how Web 2.0 technologies might foster shared interests and practices. More specifically, it would be relevant to develop rich description of how the use of Web 2.0 technologies may lead to the emergence of "new joint fields"

(Levina and Vaast, 2005) and affect the innovation development process. If the precepts of the knowledge-based perspective on innovation hold true in practice (which still need to be verified), Web 2.0 may offer great potential for innovation organizations, including software organizations. Yet for the most part, the application of Web 2.0 technologies is still something new and yet to be understood.

For the purpose of this doctoral dissertation, large business software applications developed to meet the particular needs of the bank were the innovations of interest. Because the software applications were developed for and primarily used by the bank, the focus on the research was, to some extent, inwardly directed. There certainly is a need to see how the theoretical framework can be used to study attempts to change practices in firms developing commercial, off-the-shelf software (i.e. software that is not designed for meet the needs of its maker). It would be interesting to see whether and how the logic of other fields penetrates practices of software organizations, as the corporate logic imbued IBTech. Does developing software for a certain industry or field lead software organizations to adopt certain beliefs and practices of this industry or field? If so, can this be explain in terms of symbolic power?

In a related vein, it would be highly pertinent to put the theoretical framework to the test in innovative contexts other than software development. This could be done, for example, in the field of medical research or entertainment where innovative hardware products are developed. Different contexts and different types of innovation might provide additional dimensions of similarity and contrast to be explored.

Appendix 1: IBTech's Global Organization Chart



Appendix 2: Capability Maturity Models

This appendix is intended to provide the reader with the high-level understanding of the Capability Maturity Model for Software (CMM) that is needed to understand the case presented in this doctoral dissertation.

Software process improvement

Within the realm of software engineering, 'process improvement' is a program of activities designed to improve the process capability of an organization's processes. 'Process capability' is the ability of a process to produce planned results. As the capability of each process is improved, it becomes predictable and measurable, and the most significant causes of poor quality and productivity are isolated or eliminated. By progressively improving its process capability, an organization is said to mature.

The CMM

The CMM is a framework that describes the key elements of an effective software process. The framework can be used to appraise the process capability of an organization or help an organization to develop its process capability. The CMM is not prescriptive in that it does not prescribe a specific software process: it describes "what" is to be done to increase the process capability of an organization, but does not say "how" it should be done.

History

The original concept of the framework was developed in the early 1980s by Watts Humphrey and his colleagues at IBM. Humphrey's unique insight was that software organizations had to remove impediments to continuous improvement in a specific order if they were to keep improving their processes capability overtime.

The development of the CMM formally began in 1986 as a collaboration between the Software Engineering Institute (SEI) of Carnegie-Mellon University and the U.S. federal government. The goal was to produce a framework for the U.S. federal government to assess the capabilities of its contractors in the area of software

development. The first version of the framework, released in 1991, gained rapid acceptance in the defense industry because the Department of Defense used the CMM process maturity level as an exclusion criterion for awarding many of its largest software acquisition contracts.

Maturity levels

The CMM is composed of five maturity levels (see Table 10). Each maturity level provides a layer in the foundation for continuous process improvement. At Level 1, the software process is ad hoc and chaotic. In progressing to Level 2, basic project management processes are introduced to track costs and schedule. At Level 3, the software process is documented and standardized across the organization. At Level 4, the software process is quantitatively managed and controlled. Finally, at Level 5, the software process is optimized.

Table 10: CMM for software – Maturity levels and key process areas

Maturity Level	Focus	Key Process Areas
5	Continual process improvement and optimization	Process Change Management Technology Change Management Defect Prevention
4	Product and process quality; Manage by measures	Software Quality Management Quantitative Process Management
3	Engineering processes and organizational support; Standard processes	Organization Process Focus Organization Process Definition Peer Reviews Training Program Intergroup Coordination Software Product Engineering Integrated Software Management
2	Project management processes; Tame local chaos	Requirements Management Software Project Planning Software Project Tracking and Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1	Competent people and heroics; Chaotic process	None

Key process areas

With the exception of Level 1, each maturity level is composed of several key process areas (see Table 10). Each key process area, in turn, is composed of a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability at that maturity level. By developing its process capability in the key process areas corresponding to a maturity level, an organization develops its overall process capability.

The CMMI

Since its introduction, the framework has spread across industries and has achieved significant penetration into commercial IT. Since 2001, however, the Capability Maturity Model Integration (CMMI) has progressively replaced the CMM. The CMMI encompasses the CMM, but possesses two additional disciplines – 'supplier evaluation' and 'contract monitoring.' IBTech used the two models, and in order to facilitate further discussion, the term 'CMM/I' is used in the present doctoral thesis to refer to the two models without distinction.

Appendix 3: Agile Software Development

Agile software development evolved in the mid 1990s as part of a reaction against process- and document-centric methods. In 2001, prominent members of the software community created the "Manifesto for Agile Software Development," which spelled out the values and principles encapsulating the essence of agile software development (www.agilemanifesto.org).

Values of agile development

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Principles behind the agile manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Methods

Popular agile methods include Scrum, Crystal Clear, Adaptive Software Development, and Dynamic Systems Development Method. It is perhaps Extreme

Programming that contributed the most to establish the popularity of agile methods. Extreme Programming was created by Kent Beck in 1996 as a way to rescue a high-profile project at automotive manufacturer. The agile off-the-shelf component acquired at IBTech is based on the Extreme Programming methodology.

9 References

Adler, P. A. and P. Adler (1994). Observational Techniques. In: N. K. Denzin and Y. S. Lincoln (ed.). Handbook of Qualitative Research. Thousand Oaks: 377-392.

Adler, P. S. (2006). "The Evolving Object of Software Development." Organization 12(3): 401-435.

Alvesson, M. and H. Willmott (1992). "On the Idea of Emancipation in Management and Organization Studies." The Academy of Management Review 17(3): 432-464.

Andersen, N. E., F. Kensing, F. Lunding, L. Mathiassen, A. Munk-Madsen, M. Rabech and P. Sorgaard (1990). Professional Systems Development: Experiences, Ideas, and Action. Prentice-Hall.

Anthony, R. (1965). Planning and Control Systems: A Framework for Analysis. Cambridge, MA, Harvard University Press.

Avgerou, C. (2000). "IT and Organizational Change: An Institutional Perspective." Information Technology & People 13(4).

Avgerou, C. and T. Cornford (1993). "A Review of the Methodologies Movement." Journal of Information Technology 5: 277-286.

Avison, D. E. and B. Fitzgerald (2003). Information Systems Development: Methodologies, Techniques, and Tools. New York, McGraw-Hill.

Avison, D. E. and B. Fitzgerald (2006). Information Systems Development: Methodologies, Techniques and Tools. London, McGraw-Hill.

Avison, D. E. and V. Taylor (1997). "Information Systems Development Methodologies: A Classification According to Problem Situation." Journal of Information Technology 12: 73-81.

Banach, R. (2007). "Formal Methods: Guest Editorial." Journal of Universal Computer Science 13(5).

Barrett, M. and G. Walsham (2004). Making Contribution from Interpretive Case Studies: Examining Processes of Construction and Use. Relevant Theory and Informed Practice: Looking Forward from a 20-year Perspective on IS Research: IFIP TC8 WG8.2 20th Year Retrospective, Manchester, England, Dordrecht: Kluwer.

Baskerville, R. (1991). Practitioner Autonomy and the Bias of Methods and Tools. In: H. E. Nissen, H. K. Klein and R. A. Hirschheim (ed.). Information

Systems research: Contemporary Approaches & Emergent Traditions. Amsterdam, North-Holland: 673-698.

Baskerville, R. and M. D. Myers (2004). "Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice-Foreword." MIS Quarterly 28(3): 329-225.

Baskerville, R., J. Travis and D. Truex (1992). Systems without Methods: The Impact of New Technologies on Information Systems Development Projects. In: K. E. Kendall, K. Lyytinen and J. I. DeGross (ed.). The Impact of Computer Supported Technologies on Information Systems Development. North-Holland, Amsterdam: 241-269.

Baskerville, R. L. and A. T. Wood-Harper (1996). "A Critical Perspective on Action Research as a Method for Information Systems Research." Journal of Information Technology 11: 235-246.

BBC (2006). Graduate Job Market 'on the Rise'. Accessed 12 January 2006. http://news.bbc.co.uk/1/hi/education/4603420.stm.

Bechky, B. A. (2003). "Sharing Meaning Across Occupational Communities: The Transformation of Understanding on a Product Floor." Organization Science 14(3): 312-330.

Benbasat, I., D. K. Goldstein and M. Mead (1987). "The Case Reseach Strategy in Studies of Information Systems." Management Information Systems Quarterly 11(3): 369-386.

Berger, P. L. and T. Luckmann (1967). The Social Construction of Reality. London, Allen Lane The Penguin Press.

Besser, H. (1995). The Information SuperHighway: Social and Cultural Impact. In: I. Boal (ed.). Resisting the Virtual Life: The Culture and Politics of Information. San Francisco, City Lights Press.

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

Boehm, B. and R. Turner (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Boston MA, Addison Wesley.

Boehm, B. W. (1973). "Software and Its Impact: A Quantitative Assessment." Datamation 19(5): 48-59.

Boehm, B. W. (1976). "Software Engineering." IEEE Transactions on Computers C-25(12): 1226-41.

Boland, R. J. (1991). Information System Use as a Hermeneutic Process. In: H.-E. Nissen, H. K. Klein and R. A. Hirschheim (ed.). Information Systems

Research: Contemporary Approaches and Emergent Traditions. Amsterdam, Elsevier: 439-464.

Booch, G. (1994). Object-Oriented Analysis and Design with Application. Redwood City, CA, Benjamin/Cummings.

Bourdieu, P. (1977). Outline of a Theory of Practice. Cambridge, Cambridge University Press.

Bourdieu, P. (1981). Men and Machine. In: K. Knorr-Cetina and A. V. Cicourel (ed.). The Micro-Sociological Challenge of Macro-Sociology: Towards a Reconstruction of Social Theory and Methodology. London, Routledge: 304-317.

Bourdieu, P. (1984). Distinction: A Social Critique of the Judgement of Tastes. London, Routledge.

Bourdieu, P. (1989). "Social Space and Symbolic Power." Sociological Theory 7(1).

Bourdieu, P. (1990). The Logic of Practice. Cambridge, Polity.

Bourdieu, P. (1991). Language and Symbolic Power. Cambridge MA, Harvard University Press.

Bourdieu, P. (1992). An Invitation to Reflexive Sociology. Oxford, Polity Press.

Bourdieu, P. and J.-C. Passeron (1977). Reproduction in Education, Society and Culture. London, Sage.

Bourdieu, P. and L. J. D. Wacquant (1989). "Towards a Reflective Sociology: A Workshop with Pierre Bourdieu." Sociological Theory 7: 26-63.

Briand, L., E. Arisholm, S. Counsell, F. Houdek and P. Thevenod-Fosse (1999). "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions." Empirical Software Engineering: An International Journal 51: 245-273.

Brooks, F. (1975). The Mythical Man-Month: Essays on Software Engineering. New York, Addison-Wesley.

Brown, J. S. and P. Duguid (1991). "Organizational Learning and Communities of Practice: Toward a Unified View of Working, Learning and Innovation." Organization Science 2: 40-57.

Burrell, G. and G. Morgan (1979). Sociological Paradigms and Organisational Analysis. Hants, Ashgate.

Campbell-Kelly, M. (2003). From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry. Cambridge, MA, MIT Press.

Carlile, P. R. (2002). "A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development." Organization Science 13(4): 442-455.

Carlile, P. R. (2004). "Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries." Organization Science 15(5): 555-568.

Cavaye, A. L. M. (1996). "Case Study Research: A Multi-Faceted Research Approcah for IS." Information Systems Journal 6: 227-242.

Cerruzi, P. E. (1998). A History of Modern Computing. Cambridge, MA, MIT Press.

Checkland, P. (1981). Systems Thinking, Systems Practice. New York, John Wiley & Sons.

Chernow, R. (1986). The House of Morgan: An American Banking Dynasty and the Rise of Modern Finance. Boston, Harvard Business School Press.

Chrissis, M. B., M. Konrad and S. Shrum (2003). CMMI: Guidelines for Process Integration and Product Improvement. Boston, MA, Addison-Wesley.

Chua, W. F. (1986). "Radical Development in Accounting Thought." The Accounting Review 61: 601-632.

Coad, P. and E. Yourdon (1991). Object-Oriented Analysis. Englewood Cliffs, NJ, Yourdon Press.

Cockburn, A. and J. A. Highsmith (2001). "Agile Software Development: The People Factor." IEEE Software 34(11): 131-133.

Collins, R. (1981). Cultural Capitalism and Symbolic Violence. In: (ed.). Sociology Since Mid-Century: Essays in Theory of Cumulation. New York, Academic Press.

Constantine, L. (1993). "Work Organization: Paradigms for Project Management and Organization." Communications of the ACM 36(10).

Contu, A., C. Grey and A. Ortenblad (2003). "Against Learning." Human Relations 56(8): 931-951.

Contu, A. and H. Willmott (2003). "Re-embedding Situatedness: The Importance of Power Relations in Learning." Organization Science 14(3): 283-296.

Creswell, J. (1998). Qualitative Inquiry and Research Design: Choosing Among Five Traditions. Thousand Oaks, California, Sage Publications.

Cusumano, M. A. (1989). "The Application Software Factory: A Historical Interpretation." IEEE Software March: 23-30.

Cusumano, M. A. (1991). Japan's Software Factories: A Challenge to U.S. Management. Oxford, Oxford University Press.

Cusumano, M. A. (1998). Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People. New York, Free Press.

Cusumano, M. A. (2004). The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad. New York, Free Press.

Cusumano, M. A. and R. W. Selby (1997). "How Microsoft Builds Software." Communications of the ACM 40(6): 53-61.

Czarniawska, B. (1992). Exploring Complex Organizations: A Cultural Perspective. London, Sage.

Darke, P., G. Shanks and M. Broadbent (1998). "Successfully Completing Case Study Research: Combining Rigour, Relevance and Pragmatism." Information Systems Journal 8: 273-289.

De Marco, T. (1978). Structured Analysis and System Specification. New York, Yourdon Press.

Denzin, N. K. and Y. S. Lincoln (1994). Entering the Field of Quantitative Research. In: N. K. Denzin and Y. S. Lincoln (ed.). Handbook of Qualitative Research. Thousand Oaks, CA, Sage Publication: 1-17.

DiBona, C., S. Ockman and M. Stone (2000). Open Sources: Voices from the Open Source Revolution. Sebastopol, CA, O'Reilly & Associates.

Dijkstra, E. (1968). "Go To Statement Considered Harmful." Communications of the ACM 11(3): 147–148.

Dijkstra, E. W. (1969). Structured Programming. Software Engineering: Concepts and Techniques, Rome, Petrocelli/Charter.

DiMaggio, P. J. (1979). "Review Essay on Pierre Bourdieu." American Journal of Sociology 84(6): 1460-1474.

Dubé, L. (1998). "Teams in Packaged Software Development: The Software Corp. Experience." Information Technology and People 11(1): 36-61.

Dubé, L. and D. Robey (1999). "Software Stories: Three Cultural Perspectives on the Organizational Context of Software Development Practices." Accounting Management and Information Technologies 9(4): 223-259.

Dyer, J. H. and H. Singh (1998). "The Relational View: Cooperative Strategy and Sources of Interorganizational Competitive Advantages." Academy of Management Review 23(4): 660-679.

Earl, M. J. and D. J. Skyrme (1992). "Hybrid Managers - What Do we Know about them?" Journal of Information Systems 2: 169-187.

Eisenhardt, K. M. (1989). "Building Theories from Case Study Research." Academy of Management Review 14(4): 532-550.

Elster, J. (1990). Merton's Functionalism and the Unintended Consequences of Action. In: J. Clark, C. Modgil and S. Modgil (ed.). Robert Merton: Consensus and Controversy. London, Falmer Press.

Fairclough, N. (1989). Language and Ideology. London, Longman.

Fairclough, N. (1995). Critical Discourse Analysis: The Critical Study of Language. London, Longman.

Fairclough, N. (2005). "Discourse Analysis in Organization Studies: The Case for Critical Realism." Organization Studies 26: 915 - 939.

Fairclough, N. and R. Wodak (1997). Critical Discourse Analysis. In: T. A. V. Dijk (ed.). Discourse as Social Interaction. London, Sage. 2: Discourse Studies - A Multidisciplinary Introduction: 258-284.

Fish, S. (1980). Is There a Text in This Class? Cambridge, MA, Harvard University Press.

Foucault, M. (1972). The Archaeology of Knowledge. London, Tavistock.

Fowler, M. (2003). The New Methodology. Accessed 12 January 2007. http://www.martinfowler.com/articles/newMethodology.html#N101B3.

Fox, S. (2000). "Communities of Practice, Foucault and Actor-Network Theory." Journal of Management Studies 37(6): 853-867.

Friedman, A. L. (1989). Computer Systems Development: History, Organization and Implementation. New York, Wiley.

Gadamer, H.-G. (1977). Philosophical Hermeneutics. Los Angeles, University of California Press.

Galliers, R. D. (1992). Choosing Information Systems Research Approaches. In: R. D. Galliers (ed.). Information Systems Research: Issues, Methods and Practice Guidelines. Oxford, Blackwell Scientific Publication: 144-162.

Garfinkel, H. (1967). Studies in Ethnomethodology. Englewood Cliffs, NJ, Prentice-Hall.

Gasson, S. (1999). "A Social Action Model of Situated Information Systems Design." Data Base 30(2): 82-97.

Gates, B. (1995). The Road Ahead. New York, Penguin Books.

Ghoshal, S. and C. Barlett (1988). "Creation, Adoption, and Diffusion of Innovations by Subsidiaries of Multinational Corporation." Journal of International Business Studies 19(365-388).

Giroux, H. (1983). Theory and Resistance in Education: A Pedagogy for the Opposition. New York, Begin and Garvey.

Glass, R. L. (2006). "The Standish Report: Does it Really describe a Software Crisis." Communications of the ACM 49(8): 15-16.

Grant, D. and C. Hardy (2003). "Struggles with Organizational Discourse." Organization Studies 25(1): 5-13.

Grant, D., T. Keenoy and C. Oswick (1998). Introduction: Organizational Discourses: Of diversity, Dichotomy and Multi-disciplinary. In: D. Grant, T. Keenoy and C. Oswick (ed.). Discourse and Organization. London, Sage: 1-13.

Guba, E. G. and Y. S. Lincoln (1994). Competing Paradigms in Qualitative Research. In: N. K. Denzin and Y. S. Lincoln (ed.). Handbook of Qualitative Research. Thousand Oaks, Sage.

Habermas, J. (1980). The Hermeneutic Claim to Universality. In: J. Bleicher (ed.). Contemporary Hermeneutics. London, Routledge Kegan Paul.

Habermas, J. (1984). The Theory of Communicative Action - Reason and the Rationalization of Society. Boston, MA, Beacon Press.

Habermas, J. (1987). The Theory of communicative Action - Volume Two: Lifeworld and Systems: A Critique of Functionalist Reason. Boston, MA, Beacon Press.

Hammer, M. and J. Champy (1993). Reengineering the Corporation: A Manifesto for Business Revolution. New York, Harper Business.

Hardy, C. (2004). "Scaling Up and Bearing Down in Discourse Analysis: Questions Regarding Textual Agencies and their Context." Organization 11(3): 415-425.

Hardy, C., T. B. Lawrence and D. Grant (2005). "Discourse and Collaboration: The Role of Conversations and Collective Identity." Academy of Management Review 30(1): 58-77.

Hardy, C., I. Palmer and N. Phillips (2000). "Discourse as a Strategic Resource." Human Relations 53(9): 1227-1248.

Harter, D. E., M. S. Krishnan and S. A. Slaughter (2000). "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development." Management Science 46(4): 451-466.

Heritage, J. (1984). Garfinkel and Ethnomethodology. Cambridge, Polity Press.

Highsmith, J. A. (1997). "Messy, Exciting, and Anxiety-ridden: Adaptive Software Development." American Programmer X(1).

Highsmith, J. A. (2002). Agile Software Development Ecosystems. Boston, MA, Addison-Wesley.

Hirschheim, R. and H. K. Klein (2000). Information Systems Research at the Crossroads: External Versus Internal Views. In: R. Baskerville, J. Stage and J. DeGross (ed.). Organizational and Social Perspectives on Information Technology (pp. 233-254). Boston: Kluwer Academic. Boston, Kluwer Academic: 233-254.

Hirschheim, R. A. and H. K. Klein (1989). "Four Paradigms of Information Sytems Development." Communications of the ACM 32(10): 1199-1214.

Hirschheim, R. A., H. K. Klein and K. Lyytinen (1996). "Exploring the Intellectual Structures of Information Systems Development: A Social Action Theoretic Analysis." Accounting Management and Information Technologies 6(1/2): 1-64.

Holmwood, J. (2005). Functionalism and its Critics. In: A. Harrington (ed.). Modern Social Theory: An Introduction. Oxford, Oxford University Press: 87-109.

Hussey, J. and R. Hussey (1997). Business Research. Houndmills, Palgrave.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology, Institute of Electrical and Electronics Engineers.

Iivary, J., R. A. Hirschheim and H. K. Klein (1998). "A Paradigmatic Contrasting Development Approaches and Methodologies." Information Systems Research 9(2).

Iivary, J. and M. Huisman (2007). "The Relationship Between Organizational Culture and the Deployment of Systems Development Methodologies." MIS Quarterly 30(1): 35-58.

Introna, L. D. (1996). "Notes on Ateleological Information Systems Development." Information Technology & People 9(4): 20-39.

Iversen, J. and L. Mathiassen (2003). "Cultivation and Engineering of a Software Metrics Program." Information Systems Journal 13: 3-19.

Jackson, M. A. (1975). Principles of Program Design. London, Academic.

Jackson, M. A. (1983). System Development. New York, Prentice-Hall.

Jacobson, I., M. Christerson, P. Jonsson and G. Overgaard (1995). Object-Oriented Software Engineering: A Use Case Driven Approach. Wokingham, England, Addison-Wesley.

Jenkins, R. (1982). "Pierre Bourdieu and the Reproduction of Determinism." Sociology 16(2): 270-281.

Jenkins, R. (2001). Pierre Bourdieu. New York, Routledge.

Jermier, J. M. (1998). "Introduction: Critical Perspective on Organizational Control." Administrative Science Quarterly 43: 235-256.

Johnson, J. (1991). The Software Factory: Managing Software Development and Maintenance. Wellesley, MA, QED Information Sciences.

Johnson, R. A. (2002). "Object-oriented systems Development." Communications of the Association for Information Systems 8: 65-81.

Jørgensen, M. and K. Molokken-Ostvold (2006). "How Large Are Software Cost Overruns? A Review of the 1994 CHAOS Report." Information and Software Technology 48(4): 297-301.

Kanter, R. M. (1988). "When a Thousand Flowers Bloom: Structural, Collective, and Social Condition for Innovation in Organization." Research in Organizational Behavior 10: 169-211.

Kaplan, B. and J. A. Maxwell (1994). Qualitative Research Methods for Evaluating Computer Information Systems. In: J. G. Anderson, C. E. Aydin and S. J. Jay (ed.). Evaluating Health Care Information Systems: Methods and Applications. Thousand Oaks, CA, Sage: 45-68.

Keil-Slawik, R. (1996). History and Identity. History of Software Engineering, Paderborn.

Kets de Vries, M. F. R. and D. Miller (1987). "Interpreting Organizational Texts." Journal of Management Studies 24(3): 233-247.

King, S. (1996). "Case Tools and Organizational Action." Information Systems Journal 6: 173-194.

Klein, H. K. and M. D. Myers (1999). "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems." MIS Quarterly 23(1): 67-94.

Knorr-Cetina, K. (1981). The Micro-Sociological Challenge of Macro-Sociology: Towards a Reconstruction of Social Theory and Methodology. In: K. Knorr-Cetina and A. V. Cicourel (ed.). Advances in Social Theory and Methodology. London, Routledge: 1-47.

Knuth, D. (1974). "Structured Programming with Goto Statements." Computing Surveys 6(4): 261-301.

Krishnan, M. S., C. H. Kriebel, S. Kekre and T. Mukhopadhyay (2000). "An Empirical Analysis of Productivity and Quality in Software Products." Management Science 46(6): 745-759.

Lanzara (1983). The Design Process: Frames, Metaphors and Games. In: C. Ciborra and L. Schneider (ed.). Systems Design for, with and by the Users. Amsterdam, North-Holland Publishing Company.

Latour, B. and S. Woolgar (1986). Laboratory Life: The Construction of Scientific Facts. Princeton, NJ, Princeton University Press.

Lave, J. and W. Wenger (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge, Cambridge University Press.

Lee, A. S. (1991). "Integrating Positivist and Interpretive Approaches to Organizational Research." Organization Science 2(4): 342-365.

Lee, A. S. (1994). "Electronic Mail as a Medium for Rich Communication: An Empirical Investigation Using Hermeneutic Interpretation." MIS Quarterly 18(2): 143-157.

Lee, A. S. (1999). "Rigor and Relevance in MIS Research: Beyond the Approach of Positivism Alone." MIS Quarterly 23(1).

Lee, A. S. and R. L. Baskerville (2003). "Generalizing Generalizability in Information Systems Research." Information Systems Research 14(3): 221-243.

Levina, N. and E. Vaast (2005). "The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems." MIS Quarterly 29(2): 335-363.

Levina, N. and E. Vaast (2008). "Innovating or Doing as Told? Status Differences and Overlapping Boundaries in Offshore Collaboration." MIS Quarterly 32(2): 307-332.

MacCormack, A. (2001). "Product-Development Practices that Work: How Internet Companies Build Software." MIT Sloan Management Review Winter: 75-84.

MacCormack, A. and K. Herman (2000). Microsoft Office 2000. Harvard Business School Case. Boston, MA, Harvard Business School.

MacCormack, A., R. Verganti and M. Iansiti (2001). "Developing Products on Internet Time: The Anatomy of a Flexible Development Process." Management Science 47(1).

Madsen, S., K. Kautz and R. Vidgen (2006). "A Framework for Understanding How a Unique and Local IS Development Method Emerges in Practice." European Journal of Information Systems 15: 225-238.

Mahoney, M. S. (2004). "Finding a History for Software Engineering." Annals of the History of Computing 26(1): 8-19.

Markus, L. (1983). "Power, Politics and IS Implementation." Communications of the ACM 26(6).

Marx, K. (1867). Capital I. Moscow, Progress Publishers.

Mathiassen, L. (1996). "Information Systems Development: Reflections on a Discipline." Accounting Management and Information Technologies 6(12): 127-132.

Mathiassen, L. (1998). "Reflective Systems Development." Scandinavian Journal of Information Systems 10(1 & 2).

Maynard, D. (1989). "On the Ethnography and Analysis of Discourse in Institutional Settings." Perspectives on Social Problems 1: 127-146.

McBreen, P. (2001). Software Craftsmanship: The New Imperative. New York, Addison-Wesley.

McCraw, T. K. (1997). Creating Modern Capitalism: How Entrepreneurs, Companies, and Countries Triumphed in Three Industrial Revolutions. Cambridge, MA, Harvard University Press.

McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement. Pittsburgh, PA, The Software Engineering Institute, Carnegie Mellon University.

McIlroy, M. D. (1968). Mass Produced Software Components. Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, Scientific Affairs Division, NATO.

Metiu, A. (2006). "Owning the Code: Status Closure in Distributed Groups." Organization Science 17(4): 418-435.

Meyer, J. W. and B. Rowan (1977). "Institutionalized Organizations: Formal Structure as Myth and Ceremony." American Journal of Sociology 83: 364-385.

Miles, M. B. and A. M. Huberman (1994). Qualitative Data Analysis: An Expanded Sourcebook. Thousand Oaks, CA, Sage.

Mintzberg, H. (1979). The Structuring of Organizations: A Synthesis of the Research. Englewood Cliffs, Prentice Hall.

Mintzberg, H. (1983). A Typology of Organization Structure. In: D. Miller and P. Friesen (ed.). Organizations: A Quantum View, Prentice Hall.

Mintzberg, H. (1984). "The Organization as Political Arena." Journal of Management Studies 22(2): 134-154.

Mumby, D. K. and R. Clair (1997). Organizational Discourse. In: T. A. VanDijk (ed.). Discourse as Structure and Process. London, Sage: 181-205.

Mumford, E. (1983). Designing Human Systems for New Technology: The ETHICS Method. Manchester, Manchester Business School Press.

Mumford, E. and M. Weir (1979). Computer Systems in Work Design: The ETHICS Method. New York, NY, John Wiley & Sons.

Myers, M. D. (1995). "Dialectical Hermeneutics: A Theoretical Framework for the Implementation of Information Systems." Information Systems Journal 5(1): 51-70.

Nandhakumar, J. and D. E. Avison (1999). "The Fiction of Methodological Development: A Field Study of Information Systems Development." Information Technology & People 12(2): 176-191.

Nandhakumar, J. and M. Jones (1997). "Too Close for Comfort? Distance and Engagement in Interpretive Information Systems Research." Information Systems Journal(7): 109-131.

NATO Science Committee (1968). Software Engineering: Concepts and Techniques. Proceedings of the NATO Conferences. Software Engineering: Concepts and Techniques., Garmisch, Germany.

Naur, P. (1985). Intuition in Software Development. In: H. Ehrig, C. Floyd, M. Nivat and J. Thatcher (ed.). Formal Methods and Software Development. New York, Springer.

Newell, S., M. Robertson and J. Swan (2006). Interactive Innovation Processes and the Problems of Managing Knowledge. In: B. Renzl, K. Matzler and H. Hinterhuber (ed.). The Future of Knowledge Management. Basingstoke, Palgrave Macmillan: 115-136.

Ngwenyama, O. and P. A. Nielsen (2003). "Competing Values in Software Process Improvement: An Assumption Analysis of CMM From an Organizational Culture Perspective." IEEE Transactions on Engineering Management 50(1): 100-112.

Orlikowski, W. J. (1991). "Integrated Information Environment or Matrix of Control? The Contradictory Implications of Information Technology." Accounting Management and Information Technologies 1(1): 9-42.

Orlikowski, W. J. (2002). "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing." Organization Science 13(3): 249–273.

Orlikowski, W. J. and S. R. Barley (2001). "Technology and Institution: What can Research on Information Technology and Research on Organizations Learn from Each Other." MIS Quarterly 25(2): 145-165.

Orlikowski, W. J. and J. J. Baroudi (1991). "Studying Information Technology in Organizations: Research Approaches and Assumptions." Information Systems Research 2(1): 1-28.

Oswick, C., T. Keenoy and D. Grant (2000). "Discourse, Organizations and Organizing: Concepts Objects and Subjects." Human Relations 53(9): 1115-1123.

Parker, I. (1992). Discourse Dynamics: Critical Analysis for Social and Individual Psychology. London, Routledge.

Parker, I., Ed. (1998). Social Constructionism, Discourse and Realism. London, Sage Publications.

Parnas, D. L. and P. C. Clements (1986). "A Rational Design Process: How and Why to Fake it." IEEE Transactions on Software Engineering 12(2).

Parsons, T. (1951). The Social System. London, Routledge.

Parsons, T. (1961). Theories of Society: Foundations of Modern Sociological Theory. New York, Free Press.

Paulk, M. C., B. Curtis, M. B. Chrissis and C. V. Weber (1993). Capability Maturity Model for Software, Version 1.1. Pittsburgh, PA.

Peizer, J. (2006). The Great Software Debate: Technology and Ideology, Internaut Consulting. 2006.

Peters, T. and R. Waterman (1982). In Search of Excellence. New York, Harper and Row Publishers.

Pettigrew, A. M. (1979). "On Studying Organization Cultures." Administrative Science Quarterly 24: 570-581.

Pfeffer, J. (1981). Power in Organization. Marshfield, MA, Pitman.

Pfeffer, J. (1993). "Barriers to the Advance of Organizational Science: Paradigm Development as an Independent Variable." Academy of Management Review 18(4): 599-620.

Phillips, N. and C. Hardy (1997). "Managing Multiple Identities: Discourse, Legitimacy and Resources in the UK Refugee System." Organization 4(2): 159-185.

Phillips, N. and C. Hardy (2002). Understanding Discourse Analysis: Investigating Processes of Social Construction. Thousand Oaks, CA, Sage.

Pugh, E. W., L. R. Johnson and J. H. Palmer (1991). IBM's 360 and Early 370 Systems. Cambridge, MA, MIT Press.

Putnam, L. and F. Cooren (2004). "Alternative Perspectives on the Role of Text and Agency in Constituting Organizations." Organization 11(3): 323-333.

Quinn, J. B. (1985). "Managing Innovation: Controlled Chaos." Harvard Business Review 63(July-August): 73-84.

Rapoport, R. N. (1970). "Three Dilemmas in Action Research." Human Relations 23(4): 499-513.

Raymond, E. S. (1998). Homesteading the Noosphere, Retrieved 19 April 2006 from http://www.catb.org/~esr/writings/homesteading/homesteading/.

Raymond, E. S. (2000). The Cathedral and the Bazaar, Retrieved 19 September 2004 from http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html.

Reeves Saunday, P. (1979). "The Ethnographic Paradigm." Administrative Science Quarterly December(4): 527-538.

Ricoeur, P. (1974). The Conflict of Interpretations: Essays in Hermeneutics. Evanston, IL, Northwestern University Press.

Ricoeur, P. (1981). Hermeneutics and the Human Sciences. Cambridge, Cambridge University Press.

Ricoeur, P. (1991). From Text to Action. Evanston, IL, Northwestern University Press.

Ritzer, G. (1996). Sociological Theory. New York, McGraw-Hill.

Ritzer, G. (2000). The McDonaldization of Society. Thousand Oaks, CA, Pine Forge.

Roberts, L. M. (2005). "Changing Faces: Professional Image Construction in Diverse Organizational Settings." Academy of Management Review 30(4): 685-711.

Robles, G. (2004). A Software Engineering Approach to Libre Software, Retrieved 20 April 2005 from http://www.opensourcejahrbuch.de/2004/pdfs/III-3-Robles.pdf.

Robson, C. (1993). Real World Research. Oxford, Blackwell.

Ronkko, K., O. Lindeberg and Y. Dittrich (2002). Are Software Engineering and Ethnographic Discourses Incompatible? International Symposium on Empirical Software Engineering (ISESE 2002), Nara, Japan.

Rorty, R. (1967). The Linguistic Turn. Chicago, The Chicago University Press.

Rovik, K. A. (2002). The Secrets of the Winners: Management Ideas that Flows. In: K. Shalin-Andersson and L. Engwall (ed.). The Expansion of Management Knowledge. Stanford, CA, Stanford University Press: 113-144.

Roy, W. G. (1997). Socializing Capital: The Rise of the Large Industrial Corporation in America. Princeton, NJ, Princeton University Press.

Russo, N. and E. Stolterman (2000a). "Exploring the Assumptions Underlying Information Systems Methodologies: Their impact on Past, Present and Future ISM Research." Information Technology and People 13(4).

Russo, N. and E. Stolterman (2000b). Identifying Assumptions Underlying Information Systems Methodologies. In: R. Hackney and D. Dunn (ed.). Business Information Technology Management: Alternative and Adaptive Futures. New York, Macmillan Publishing.

Sacks, H., E. Schegloff and G. Jefferson (1974). "A Simplest Systematics for the Organization of Turn-taking for Conversations." Language 50(4): 696-735.

Sahlin-Andersson, K. and L. Engwall (2002a). Carriers, Flows, and Sources of Management Knowledge. In: K. Shalin-Andersson and L. Engwall (ed.). The Expansion of Management Knowledge. Stanford, Stanford University Press: 3-32.

Sahlin-Andersson, K. and L. Engwall, Eds. (2002b). The Expansion of Management Knowledge: Carriers, Flows, and Sources. Stanford, CA, Stanford University Press.

Sauer, C. and C. Cuthbertson (2003). The State of IT Project Management in the UK 2002-2003. Oxford, University of Oxford.

Scott, R. W. (2003). "Institutional Carriers: Reviewing Modes of Transporting Ideas Over Time and Space and considering Consequences." Industrial and Corporate Change 12(4): 879-894.

Scott, W. R. (2001). Institutions and Organizations. Thousand Oaks, CA, Sage.

Selby, R. W. (2005). "Enabling Reuse-Based Software Development of Large-Scale Systems." IEEE Transactions on Software Engineering 31(6).

Senge, P. (1990). The Fifth Discipline: The Art and Practice of the Learning Organization. New York, Doubleday.

Serour, M. K. and B. Henderson-Sellers (2002). The Role of Organizational Culture on the Adoption and Diffusion of Software Engineering Process: An Empirical Study. Proceedings of the IFIP WG8.6 Fifth International Working Conference on the Adoption and Diffusion of IT in an Environment of Critical Change. August 1-3 2002, Sydney, Australia, Pearson Publishing.

Sewell, G. (1998). "The Discipline of Teams: the Control of Team-based Industrial Work Through Electronic and Peer Surveillance." Administrative Science Quarterly 43: 397-428.

Silverman, D. (1998). "Qualitative Research: Meaning or Practices?" Information Systems Journal 8: 3-20.

Simon, H. (1957). Models of Man. New York, Wiley.

Simon, H. and J. G. March (1958). Organizations. New York, John Wiley & Sons.

Smith, H. A. and J. D. McKenn (1996). "Object-Oriented Technology: Getting Beyond the Hype." The Data Base for Advances in Information Systems 27(2): 20-29.

Stolterman, E. (1991). "How System Designers Think About Design and Methods." Scandinavian Journal of Information Systems 3: 137-150.

Suchman, L. (1987). Plans and Situated Actions: The Problem of Human/Machine Communication. Cambridge, Cambridge University Press.

Swan, J. and S. Newell (2000). Linking Knowledge and Innovation. Proceedings of the European Conference on Information Systems, Vienna.

Swanson, B. E. and N. C. Ramiller (1997). "The Organizing Vision in Information Systems Innovation." Organization Science 18(5): 458-474.

Swanson, K., D. McComb, J. Smith and D. McCubbrey (1991). "The Application Software Factory: Applying Total Quality Techniques to Systems Development." MIS Quarterly December: 567-579.

Szulanski, G. (1996). "Exploring Internal Stickiness: Impediments to the Transfer of Best Practice within the Firm." Strategic Management Journal 17(Winter): 27-43.

Szulanski, G. (2000). "The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness." Organizational Behavior and Human Decision Processes 82(1): 9-27.

The Standish Group (1994). The CHAOS Report. Boston, MA.

Thompson, J. B. (1990). Ideology and Modern Culture: Critical Social Theory in the Era of Mass Communication. Stanford, CA, Stanford University Press.

Thompson, J. B. (1991). Editor's Introduction. In: J. B. Thompson (ed.). Language and Symbolic Power. Cambridge, Blackwell: 1-31.

Truex, D., R. Baskerville and J. Travis (2000). "Amethodical Systems Development: The Deferrred Meaning of Systems Development Method." Accounting Management and Information Technologies 10: 53-79.

Tushman, M. L. and D. Nadler (1986). "Organizing for Innovation." California Management Review 28(Spring): 74-92.

Van de Ven, A. H. (1986). "Central Problem in the Management of Innovation." Management Science 32: 590-607.

Van Maanen, J. (1979). "The Fact of Fiction in Organization Ethnography." Administrative Science Quarterly 24(4): 539-550.

Van Maanen, J. (1988). Tales of the Field: On Writing Ethnography. Chicago, University of Chicago Press.

Walsham, G. (1993). Interpreting Information Systems in Organizations. Chichester, Wiley.

Walsham, G. (1995). "The Emergence of Interpretivism in IS Research." Information Systems Research 6(4): 376-394.

Wasserman, A. I. (1996). "Toward a Discipline of Software Engineering." IEEE Software(November): 23-31.

Wastell, D. (1996). "The Fetish of Technique: Methodology as a Social Defence." Information Systems Journal 6(1): 25-40.

Weber, M. (1968). Economy and Society. New York, Bedminster Press.

Whetten, D. A. (1989). "What Constitute a Theoretical Contribution?" Academy of Management Review 14(4): 490-495.

Wittgenstein, L. (1953). Philosophical Investigations. Oxford, Blackwell.

Wynekoop, J. and B. Russo (1997). "Studying Information Systems Development Methodologies." Information Systems Journal 7(1): 47-65.

Yin, R. (1984). Case Study Research: Design and Methods. Berverly Hills, CA, Sage.

Yin, R. K. (1981). "The Case Study Crisis: Some Answers." Administrative Science Quarterly 26(1): 58-65.

Yourdon, E. and L. Constantine (1979). Structured Design. Englewood Cliffs, N.J., Prentice Hall.

Zucker, L. (1987). "Institutional Theories of Organization." Annual Review of Sociology 13: 443-464.