Chapter I
Organizational Analysis of Small Software Organizations: Framework and Case Study

Jesús Zavala-Ruiz
Metropolitan Autonomous University – Iztapalapa, Mexico

ABSTRACT

The intention of this chapter is twofold. On the one hand, I illustrate the complexity of the small software organization, because it is not a reduced version of a large company. Rather, it has very important advantages and challenges. Then, I use organization studies as a multi-disciplinary and multi-paradigmatic link between disciplines, able to reconcile those distinct visions. On the other hand, I open the discussion on the state of crisis affecting software engineering as a discipline. For that, I try to sensitize the reader to the facts surrounding this crisis, but also to the most promising alternative, which is the redefinition of software engineering as a discipline. One of the possible options for that paradigmatic change requires a multi-disciplinary orientation because their positivist roots and the adoption of a constructivist ontology and epistemology facilitating the inclusion of visions non-qualified for a systematic, disciplined and quantitative approach. My position is that only by opening up this discussion is it possible to begin transforming and consolidating software engineering as a strengthened and more terrain-attached discipline because of its powerful theoretical and practical explanatory capacity.

INTRODUCTION

In modern society, service-related activities are growing in importance. Software is playing a preponderant role. More and more, organizations depend on data-processing processes and on the abilities of highly skilled and scarce personnel. They have been named *knowledge workers* and this kind of organization is named *knowledge intensive firms* (Alvesson, 2004, p.5ss) or *post-modern organizations*. Software organizations are a paradigmatic case of intensive-knowledge

organizations for many reasons, but principally because its main work is symbolic or non-material in nature and because it is non-standard work for its labour conditions.

Paradoxically, most software organizations are not characterized as applying to themselves what they promote for others. A closer, detailed look inside them frequently reveals a totally different situation: while in the business, most productive, administrative and management processes are candidates for automation, in software organizations, these processes still continue to develop in a craft-like and chaotic manner. The same thing happens in the systems department of any organization. Software production and the most of the work relating to information technology have succumbed to the culture of urgency and competition typical of management. Business cycles decrease and everything is in constant change, with the rising pressure of changing information systems.

Management in software organizations, in a process of imitation, has attempted to adopt the same management recipes applied in business. In this way, it has succumbed to management fads. One of these clearly is *total quality management* (TQM), adopting the name of the normative standards of that moment. Many TQM initiatives have failed miserably in practice in small businesses as well as in large ones. However, in implementing these organizational models, organizational limits exist for their full application, most importantly in small and medium enterprises (SMEs). Resource constraints in SMEs at all levels (monetary, human, time, opportunity, etc.) make it impossible to adhere completely to such prescriptions. Additionally, those organizational characteristics, such as size, diversity, structure and formalization, impose specific restrictions since causes can be manipulated, but not effects. Possibly, the consideration of formal-informal structure duality is the most important, because it provokes both structures to distance themselves from each other, making a close attachment of the informal one to the formal one practically impossible and expensive. But this structural duality is present in all aspects of the organization and software projects become chaotic because of that, among others. To analyze this organizational complexity, which is magnified in small organizations, a framework has been developed based on organization studies and with a social constructionism approach.

This chapter is organized into three parts. In the first part, I provide an overview of the organizational paradigms that allow estimating the complexity of the organization conceived as a social construct beyond its classical mechanics and simple vision. This way, the organization appears as complex, multi-dimensional and multi-paradigmatic. Additionally, I provide an overview the three disciplines that support software production: management, project management and software engineering. I end with a recounting of facts pointing to software engineering as a discipline in crisis, and the causes of that crisis identified.

In the second part, I propose that, if software engineering is in crisis, a paradigmatic change as a discipline is required and I propose three alternatives for that change. In addition, based on Kenneth Gergen's arguments, I assume that all paradigmatic change can only be possible by means of abandoning the current paradigm, adopting a critical attitude with a new discourse and arguments, and then promoting a new paradigm. This part ends with positing the conceptual tools that constitute a framework for doing organizational analysis in software organizations. It is exemplified by a study case of a small outsourcing software company in Mexico City. In addition, I reflect briefly on the scope and utility of this framework. The application of this framework demonstrates that the use of a distinct paradigm other than software engineering for studying software organizations is possible. Finally, in the third part, I explore some of the main trends I consider promising for research in the short term.

BACKGROUND

Organization in an Organization Studies Approach

In their essay *Organizations, Organization and Organizing*, Steward R. Clegg and Cynthia Hardy (1996) give the most commonly accepted definition of *Organization Studies* as "a series of conversations," in particular those of organization-studies researchers who help to constitute organizations through terms, "derived from paradigms, methods and assumptions, themselves derived from earlier conversations" (p. 3). In other words, organization studies are "a place of meeting", "a crossroads," of disciplines and discourses as Luis Montaño Hirose observes (Montaño, 2005). Therefore, organization studies embrace disciplines like engineering, management, economics, psychology, sociology, anthropology and human sciences, among others, all revolving around organizations. Similarly, an *organization* is:

"a collective with a relatively identifiable frontier, a normative order, levels of authority, systems of communications and of coordination; this collective exists in a continuous way in an environment which is involved in activities that are related in general around a group of goals; the activities throw results for the members of the organization, the same organization and the society." (Scott, 1992, p. 25)

In an organization studies approach, according to Clegg and Hardy (1996), *organizations* are conceived as "empirical objects" (p. 3) and "sites of situated social action more or less open both to explicitly organized and formal disciplinary knowledges [...] and also to conversational practices embedded in the broad social fabric" (p. 4). In other words, organizations are social constructs depending on the participants and the approach used to study them. Two related concepts are *organization*, conceived as a "theoretical discourse," and *organizing*, as a "social process" (p. 3). It is still an on-going debate, but it shows a multi-dimensional arrangement of all organizations.

Now I'll review the most common paradigms accepted in organizations. In his essay *Organizational Theorizing: A Historically Contested Terrain*, Michael Reed (1996) has provided an instructional account of the analytical narratives and the ethical discourses shaping organization studies. The overview of organization theory has been structured using Reed's framework. It captures the complete array of theorizing to date and provides a useful guide for a more sophisticated exploration. 'Organization', 'human being' and 'management' have been chosen as central categories for the ontological characterization of each paradigm. In an epistemological point of view, the most representative theories or approaches close to each paradigm have been cited, being illustrative rather than exhaustive. In this sense, Reed parses organization theory into six analytical *metanarratives*, in a progressive and historical approach.

Reeds' (1996) first paradigm is *rationality*, illustrated with the phrase "from night watchman state to industrial state." It conceives *organization* as a *rationally constructed* artefact for resolving collective problems of social order and administrative management (p. 29). In other words, organization is conceived as a *tool* or *instrument* for pursuing collective goals through the "design and management of structures" directed toward administering and manipulating organizational behaviour (p. 30). In this rational paradigm, *human beings* are "raw material," transformed by modern organizational technologies into "well ordered, productive members of society" (p. 30). *Management* translates highly contestable, normative precepts into universal, objective, immutable and, hence, unchangeable laws. Therefore, efficiency and effectiveness permeate all, including human beings. In other words, management transforms "an intuitive craft into a codified and analyzable"

science (p. 30). Taylorism is "an important component of the philosophical outlook of modern industrial civilization" (p. 29) quite close to the rational paradigm. *Epistemology* is supported by bounded rationality, administrative behaviour and decision-making theories. Strategic analysis inscribes in this way also. This paradigm has been the most dominant in the past and at present.

Reed's (1996) second paradigm is *integration* ("from entrepreneurial capitalism to welfare capitalism"). The ontologies of this paradigm are *organization* conceived as a *social system* that facilitates the integration of individuals into society. *Human beings* are discovered as more social and psychological, that is, rational, emotional, and social rather than only rational, and *management* is reinforced by social engineering and flexible design. *Epistemology* is supported by sociotechnical systems and contingency theories, among others, always in a structural-functionalist and systems approach (pp. 31-32).

Reeds' (1996) third paradigm is *market* ("from managerial capitalism to neo-liberal capitalism"). This paradigm conceptualizes *organization* as a unitary social and moral order in which individual and group interests are fitted and lead by suprahuman and live market forces. *Human beings* are cooperative agents rewarded by material, economic and moral incentives. *Management* leads invisible market forces oriented to visible organizing. *Epistemology* uses agency and firm theories grounded in this paradigm (pp. 33-34).

Reeds' (1996) fourth paradigm is *power* ("from liberal collectivism to bargained corporativism"). In this paradigm, *organization* is conceived as a place of "conflicting interests and values constituted through power struggle" and power is responsible for success and failures in organizations. *Human beings* are immersed in the domination logic as dominant or as dominated, that is, involved in a power game. *Management* is based on episodic, manipulative and hegemonic power using institutional and political discourses. *Epistemology* utilizes Weberian and Foucauldian approaches widely, and labour-process theory and total quality management emerged from this paradigm (pp. 34-35).

Reeds' (1996) fifth paradigm is *knowledge* ("from industrialism/modernity to post-industrialism/post-modernity") that conceives ontologies as follows: *organization* as "a socially constructed and sustained 'order' necessarily grounded in the localized stocks of knowledge, practical routines and technical devices" (p. 36), that is, as result of "the condensation of local cultures of values, power, rules, discretion and paradox" (Clegg, cited by Reed, 1996, p. 36). The *glocality* (impact globally, act locally) is a common discourse. *Human beings* are the new *skilled workers,* empowered by knowledge but in conflict with management (e.g., computer programmer). *Management* is faced with the challenge of managing empowered knowledge workers as experts without destroying their innovative initiatives in a cultural global/local environment. *Epistemology* favours ethnomethodology, postmodernist, neorationalist, actor-network and poststructuralist approaches (pp. 35-36).

Reeds' (1996) sixth paradigm is *justice* ("from repressive to participatory democracy"). Ontology in this paradigm conceives the *organization* as an institutionalized structure of power and authority surrounding the localized micropractices of organizational members (p. 37), but tending to be a paradoxical and chaotic, but ordered new post-modern organization. *Human beings* are institutionally and organizationally mobilized, but historically and culturally grounded, and professionalism is promoted. *Management* is reinforced by social and economic governance. *Epistemology* utilizes neo-institutionalism and postmodernism intended to address complexity using an inclusive, multi-disciplinary approach. Positivist/constructivist philosophical and epistemological debate occurs (p. 39).

In my opinion, Reed's (1996) framework may be extended to a seventh paradigm: the *human* paradigm. This paradigm may be expressed as

from fragmented life to integrative life. This paradigm conceives of an *organization* as a social construct and discourse, a place for economic purposes but also for social and personal growth too. In other words, *organizations* are built by human beings for human beings serving humanity, not serving against them. *Human being* includes rational, emotional, sentimental, social, objective, subjective, quantitative, and qualitative, among other complimentary human facets, all in action. *Management* is conceived as paradoxical, favouring innovation, liberty, and commitment. It abandons a fragmentary and disciplinary control paradigm, and business firms turn into real ethical and socially responsible organizations. *Epistemology* is based on multi-disciplinary human sciences.

As I illustrated in the brief review of organization theory, the complexity of the organization is evident when each paradigm is added to the previous one and the organization has been conceived as an object of study much more complex, because previous visions have not been abandoned but have been incorporated into current ones.

SOFTWARE ORGANIZATION: A POST-INDUSTRIAL FACTORY OR A POST-MODERN SHOP?

Software organization or software firm, also known ironically as *software factory*, may be conceived in different ways: (1) a *production factory* of interchangeable parts (McIlroy, 1969), or rational paradigm; (2) complex strategic management and production infrastructure *facilities* (Laporte, Papiccio, & Trudel, 1998, p. 1), or power paradigm; (3) a manufacturing, product development, and automation *flexible factory* (Griss & Wendtzel, 1994, p. 48), or market paradigm; or (4) an *experience factory* (Basili, Caldiera, & Rombach, 2002, p. 8), or knowledge paradigm. The final definitions of software organization are close to the logic of an organization as an evolution of manufacturing with mass production lines in the rational and mechanistic management paradigm, Taylorian Scientific Management, but they are simplistic because organizations are so much more than production functions alone. The rational approaches adopted only "achieve" descriptions and quantitative calculus, but they are unable to successfully address the complex organizational and management issues involved in software development, because cause-effect schemes are not so clear in practice. In other words, the software organization may be conceived better as a social construct, not a thing that is not governed by linear cause-effect laws but rather by social, non-predictive and irrational organizational behaviour (social process or social interaction) as illustrated by Reed's (1996) paradigm framework and the constructionism approach has proposed (cf. Burr, 2003, p. 1ss; Hibberd, 2005, p. 2ss).

But a clear characterization of the current software organization, adapted in a literary organizational perspective, was depicted in masterfully by Moschedai Ben-Ari (1998). He narrates an imaginary visit to an ancient twenty-first century software factory in the year 2300. The most common characteristics of interest in this discussion are as follows.

First, according to Ben-Ari (1998), the *production model* of software organization is Taylorist, that is, a disciplinary system in a wide sense oriented toward controlling employees, process and product using authoritarian bureaucratic management:

In those pioneering days of the computer industry, there was apparently a strict division between professional engineers who knew how to design and build a computer, and another class of workers who adapted these computers to specific applications. […] *They were called 'programmers' or 'software developers.'*(p. 89)

The coordinator of the factory […] *was usually called 'president' or 'manager,' which were euphe-*

mistic terms applied to someone who functioned as an authoritarian dictator of the factory. (p. 90)

Second, Ben-Ari (1998) states that software organization, frequently is immersed in the "high performance culture" based on the cheapest, disciplined, half-educated and inexperienced, but reckless, young workers. The *business model* of software organization has a high-profit, short-term orientation. The productive consequences of this are high turnover rates, skilled-worker "piracy" between firms, the overexploitation and precariousness of the labour workforce, short productive work life, deficient software quality and "mercenary" behaviour, I consider:

Recently discovered historical documents show that the workers were forced to labour for fourteen to sixteen hours a day, six or seven days a week. (Ben-Ari, 1998, p. 90)

The result of using mass [cheapest and disciplined] *labour was that the factory output was extremely high, but since it was produced by half-educated, inexperienced even reckless young workers, it was of such low quality that large numbers of workers were employed fixing flaws in the products.* (p. 91)

Third, the *personal and social impact* exhibits anomie, out of balance, fragmentation and loss of meaning in social, personal and work life (Sievers, 1994, pp. 12-46), as well as work addiction (workaholism):

Not only was this extremely detrimental to their physical health, but therapists have shown that this leads to a complete breakdown of family and community life. Divorce was rampant and children of these workers showed high levels of psychological trauma resulting from parental neglect. (Ben-Ari, 1998, p. 90)

Finally, Ben-Ari describes a *management style* centred on a culture of excellence as a management paradigm facilitating enrollment of new employees and personnel management, but generally causing stress and burn-out:

Forensic psychologists believe that many people attracted to this discipline had infantile personalities, and resisted efforts to place it on a sound economic footing. (Ben-Ari, 1998, p. 89)

One theory holds that a psychological technique called 'brainwashing' was used to convince software factory workers that labour under these conditions was spiritually rewarding and that selective admission to this prestigious profession was a mark of superiority. The theory is plausible, because the alternative that workers voluntarily accepted these conditions is outrageous. (p. 90)

Moschedar Ben-Ari's literary description of software factory, in my opinion, ties into the characteristics of the typical software organization. In some aspects, management style in software organizations is similar to the *psychic management* or *managinary management* (managinary derives from a linguistic contraction of "management" and "imaginary") that was found in France by Aubert and Gaulejac (1993, parts 2-3) and Aubert (1994). This management style gets the psychic energy from its members, until the members burn-out. It occurs in large firms, but frequently also in SMEs. Asterion, who probably was a burned-out software worker when he wrote this message in his blog, illustrates it:

"July 18, 2005
I'm tired.
I'm fed up with my work.
I'm tired of programming like this, patching and tying everything with wire. I'm fed up of everything being for yesterday, everything burning, always urgent. Of programming "as it turns out,"

discarding optimal solutions in favour of those taking less development time, without planning anything. Here [in Argentina] *the expression 'software architect' we've never heard even the name of. Everything is improvised, with more or less degree of luck.*
In our 'software factory' we are all equal. Rotten. And that's the problem. People talk bad about the company, of those leading it, there's bad vibes. There are meetings, comments, coteries. And I'm tired of that climate, tired of being hostage of those rumours of war.
We're like a defeated army, perhaps the 'wasted procession' of which John Dryden spoke.
Our work place is any old thing. It's a house, far from downtown. We are packed like sardines. In the Winter we freeze and in the Summer the opposite. There are too many people together in a very small space.
The worst thing of all is that there is no outlook of change. It's always going to be like this, per secula seculorum, ad nauseam.
All these things, though I don't want them to, affect me. They take away my desire to do things.
[…]
They sent me to work at another company, for several months.
The place is fantastic. It's a modern, spacious office, very well set up.
But what really killed me is the labour climate. People come to work in good mood, awake, and apparently happy with the destiny they were lucky to get.
I have to go. Life doesn't wait." Asterion (2005)

This burn-out phenomenon demonstrates that an organization is a complex social system in which people work, create, desire, love, enjoy and live, but also destroy, hate, suffer and "die" (psychically speaking). When workers enroll in the organization, they interweave their own stories with the story of the organization. This specificity makes each organization unique in its existence and for studying it: history, owner, leadership, entrepreneurship, management, people, production process, work, labour relations and culture, among others. There are no universal recipes for addressing that. The reader can appreciate that the multi-dimensional nature of the organization makes it very complex because it includes aspects such as economics, entrepreneurial aspects, management, work organization, coordination and communication, personnel management, relationship management, politics and sustainability, among others. Next this complexity will be analyzed.

THE COMPLEXITY OF THE SMALL IN SOFTWARE ORGANIZATION

Two important size aspects influencing the organizational complexity are the size of the firm and the size of the workgroup. Categories of *firm size* depend on the research objective but, independently of this, in studying software organizations, one must similarly focus on a measure equivalent to a full-time worker, because personnel turnover is high, among other issues. The appropriate organizational size for software firms is not yet fully addressed and requires further in-depth research. There is no universal classification of firm size. Therefore, the criterion of both the Organisation for Economic Co-operation and Development (OECD) and the European Union may be appropriate. Formally, the definition of small and medium enterprise (SME) is:

non-subsidiary, independent firms which employ fewer than a given number of employees. This number varies across countries. The most frequent upper limit designating an SME is 250 employees, as in the European Union. However some countries set the limit at 200, while the United States considers SME to include firms with fewer than 500 employees. (OECD, 2002, p. 7)

Table 1. SME firm size criteria (Source: European Commission 2005, pp. 13-14)

Enterprise category	Headcount	Turnover	Balance sheet total
Medium	< 250	≤ € 50 million	≤ € 43 million
Small	< 50	≤ € 10 million	≤ € 10 million
Micro	< 10	≤ € 2 million	≤ € 2 million

Throughout the world, SMEs represent a large percentage of enterprises (up to 99%), economic output, employment (up to 70%) and innovation share (up to 60%), and provide flexibility to the economy (OECD, 2002, pp. 7-11). The most important aspect, due to SMEs' economic vulnerability, is financing strategy during start-up, the most vulnerable stage. SME owners appeal to diverse financing sources, such as credit unions, leasing companies, as well as personal and family relations (p. 19).

Small companies differ from large ones in four major aspects: (1) *territorial scope of action and mobility* (small firms concentrate on local markets in contrast to multinational companies (MNC) because of their asymmetric power); (2) *innovation and organizational change* (small firms are important sources of innovation and have innovative advantages in several respects: they spend less on research and development because of their limited available resources); (3) *regulations* (compliance costs are relatively higher for SMEs); and (4) *capital–labour ratio* (SMEs usually require more labour-intensive work, labour productivity tends to be lower and unit labour costs are higher) (Traxler, 2005, p. 300). Due to these differences, it is comprehensible that large firms and small firms may have distinct and even conflicting interests (p. 301). Some precise observations about SME characteristics are necessary: (1) small firms are essentially different, that is, not just smaller versions of large ones (Mugler, 2004, p. 2); (2) overwhelming evidence shows that smaller companies, in general, are not inferior to larger units on economic performance and firm size (Aiginger & Tichy cited by Traxler, 2005, p. 300); (3) entrepreneurs and entrepreneurship play an important role, and entrepreneurship theory may be an alternative approach for studying SMEs (Mugler, 2004, p. 2); (4) the personality of the entrepreneur is often more critical; and (5) the owner of an SME generally will not replace himself/herself in cases of failure (Mugler, 2004, p. 5). In short, for small firms, four groups of variables play a decisive role: (1) the environment of the firm, (2) the resources of the firm, (3) the personality of the entrepreneur, and (4) the management system adopted (Mugler, 2004, p. 7).

In SMEs, strategy and decision making are wide-ranging. A business owner uses direct observation, oral communication, direct sensing from the social environment (friends, family members, key employees, or quasi-family members), market, partners, and competitors, including professional analysts and consultants (Mugler, 2004, p. 5), among others. Day-to-day activities in the organization differ greatly from those roles, stereotypes, and functions proposed by classic management, as Henry Mintzberg demonstrated. Two famous works caveat the most robust pillars of management, that is, manager's work and decision-making: *The Manager's Job: Folklore and Fact* (Mintzberg, 1975) and *Decision Making: It's Not What You Think* (Mintzberg & Westley, 2001), showing there is not universal truth on managerial roles.

There are few studies on software organizations that study firm size and performance. Carmel

and Bird (1997) made a statistical study with a sample of 74 software firms in the business of software package development. The authors classify them as: (1) *embryonic* (1-19 employees), (2) *small* (20-100 employees), and (3) *medium-to-large* (over 100 employees). These authors focused on the organization workgroup and found that only 11 of 74 cases diverge from the "small team" (1-9 employees) and only 5 diverge by more than 50% (larger than 15) (paragraph: Measures). Alfredo Hualde (2004) studied the software industry linked to the *maquila* concept and the off-shore outsourcing phenomenon in Baja California, Mexico. He used a cluster perspective and confirms the small as the most common property, with a percentage of 75% of the sample ranked five employees or less (p. 15).

With regard to *team size*, it is frequently demonstrated that *small teams* of skilled developers can develop innovative complex applications on-time and on-budget, with high levels of quality and customer satisfaction (Carmel & Bird, 1997, paragraph: Team Size & Group Size; Greenfield & Short, 2003, p. 16). Software firms depend on highly skilled individuals, highly integrated and mature teams, and highly capable managers. In Bangalore, a study determined that, in the software industry, the organization's dependency on individuals' capacity is greater and that an organization's competency in the market is determined by the nature of its workforce (Ilavarasan & Sharma, 2003, p. 6).

But, what does small mean? A team size defined as *small* is relative. According to E.F. Schumacher, who popularized the phrase "small is beautiful" in 1973, advocating small teams is a key factor in success and innovation for everything from corporate committees, government units, to technical-design activities (Carmel & Bird, 1997, paragraph: Team Size & Group Size). These authors, in an in-depth study of software firms, coined that *a work team of 10 or above is a violation of the "small is beautiful" rule*. They remark that organization structure, communications and coordination are a challenge as the organization grows, which has been confirmed by organization theory since the Woodward and Pugh studies in the fifties. Carmel and Bird found that, in all cases, the work teams studies tended to be small in size, following the "small is beautiful" rule. They considered that the best sub team structure and size has not yet been addressed, but frequently find that the rule of thumb is to "split up into sub teams of five plus/minus two" and conclude that this requires more research (Carmel & Bird, 1997, paragraph: From Size to Structure). The reason they argue for the complexity of the organization is intercommunication between team members.

Two basic team structures are common in software organizations: the *permanent team* and the *ad hoc team*. In the *permanent team*, members "are involved throughout the entire development cycle" and may even "see the product through several development cycles." Members are "empowered with decision-making responsibilities" and with "ownership of the product," as a result of members' long tenure and decision-making responsibilities. Such teams generally have a very high level of commitment. On the other hand, in an *ad hoc team* "the project manager may be one of the few people involved throughout the entire development cycle," and other team members "may be involved for short periods, or just briefly every day" like a matrix organization. Because of this flexibility, "managers can optimize personnel assignments and provide more varied and stimulating work to employees." In both permanent and ad hoc teams, the "small is beautiful" rule was observed.

Generally speaking, team boundaries were usually quite clear for smaller teams, but tend to become more ambiguous with larger ones, and these team boundaries were a function of the "style and culture of the firm, the personal proclivities of the team members and of its management" (Carmel & Bird, 1997, paragraph: From Size to Structure). However, concluding that management style and culture are responsible for non-accountability may be problematic and simplistic, I

consider. For that reason, it should be considered more carefully.

The final discussion exhibits that a work team or project may be considered as an analysis unit with the appropriate dimensioning because of the fully organizational properties it possesses. In conclusion, optimal team size and optimal organization size have not yet been fully addressed by organization theory. There is no in-depth research on SMEs, then, qualitative case studies looking for detailed descriptions, and explicative analyses are required.

ORGANIZATION DIVERSITY AS A SOURCE OF COMPLEXITY

Diversity is the main characteristic of biological and social life. It provides sustainability to the whole as a live and social ecosystem. In social science, ecological approaches to organization have been developed, too. Although it is not dominant today, the analysis of organizational diversity is based on an ecological concept: *isomorphism* as a process constrainer that forces a unit in a population to resemble other units facing the same environmental conditions (DiMaggio & Powell, 1983, p. 149). In an institutional approach, three general *mechanisms of isomorphism* are conceived: (1) *coercive*, when an organization is compelled to adopt structures or rules, (2) *mimetic*, when one organization copies another, often because of uncertainty, and (3) *normative*, when the organization adopts forms because professionals claim they are superior (pp. 150-154), but in spite of this theoretical isomorphism, in practice, there is huge organizational diversity.

DiMaggio and Powell (1983) argued for three types of *conformity mechanisms*: (1) *structural*, such as institutional rules, government regulation, environmental uncertainty, desire for legitimacy, hiring personnel from successful firms or hiring consultants, and accreditation programs, among others; (2) *procedural*, as many of the "rational myths," for example, total quality programs, PERT charts becoming standard procedure, and professional groups; and (3) *personnel*, for example, specialized roles filled by certified professionals, hiring of specific types of personnel, licensing or accreditation as "qualified" personnel, certification and educational requirements (pp. 150-154).

ORGANIZATIONAL STRUCTURE AS A SOURCE OF ORGANIZATIONAL COMPLEXITY

Many organizational typologies are used as ideal models, similar to Weberian ideal types (Heckman, 1983, pp. 121-123), for contrasting reality with its ideal model. For example, based on Burns and Stalker's studies on firms in England, organizational literature distinguishes two basic organization structures and two types of management style: *mechanistic* and *organic*. *Mechanistic organizations* are associated with the so-called rational organizations conceived as instrumental and tend to be rigid, formalized, non-specialized, routinized work, undifferentiated horizontally with more hierarchy, among other features. *Mechanistic styles* were found in more stable environments (managers broke down tasks into specializations with precise job descriptions and hierarchical interaction with management). On the other hand, *organic firms* were found in unstable systems where adaptation to their specific context is required. Job roles are redefined continuously, and workers had to perform their jobs using their own knowledge. Communication occurs more laterally than through hierarchical structures. Organic organizations are closer to knowledge organizations, consisting of more flexible, informal, highly specialized, not routinized and creative work with decentralized and flatter structures (Boje, 1999, Table 1). But mechanistic/organic structures are not unique and there is no full consensus.

The most common framework for studying organizational structure, not exempt from criticism, are the *organizational configurations* proposed by Henry Mintzberg (entrepreneurial organization, machine, professional, diversified, innovative or adhocracy, missionary, and political organization). This framework consists of six basic parts and six coordinating mechanisms (Mintzberg, 1979). Mintzberg's configurations should be considered ideal models in a Weberian sense, but empirical research gives us more insight on organizational variety. For example, Aaen, Bøtcher and Mathiassen (1997), after reviewing the organizational structures proposed for software factories based on Mintzberg's model, conclude that the appropriate organizational form for a software firm is the *professional bureaucracy*, in which professional competence is viewed as more important than standardized procedures and advanced technologies (p. 431). Ensmenger (2001b) explored professionalism as a form of predicting the organizational behaviour but his study is not conclusive. In other empirical research, Meijaard, Brand and Mosselman (2005), based on the study of a stratified sample of 1,411 small Dutch firms, found more diversity based on Mintzberg's configuration criteria: (1) entrepreneur with a "submissive" team, (2) co-working boss with an open structure, (3) entrepreneurial team, (4) boss-loose control, (5) boss-tight control, (6) singular structure, (7) u-form, (8) matrix organization, and (9) m-form (pp. 17-18). These organization configurations were constructed based on terms of three-year persistent sales growth performance.

In conclusion, organizational structure is an important feature, but this is generally associated with the formal structure of the organization and it is a partial view of that because organizations are made up of formal and informal structures. A *formal structure* is that in which the social positions and relationships between members have been explicitly specified, and positions are defined independently of the personal characteristics of the participants occupying these positions (Scott, 1992, p. 18). This structure is closer to a mechanistic or bureaucratic organization. The formal structure is formally established by means of the organizational vision, mission, chart, statutes, laws and regulations, policies, procedures and plans that establish and assign authority and resources for completing goals by projects or programs. In short, formal structure is *"how things should be."* The formal structure is useful for planning or other formal procedures, but clearly it does not correspond to reality.

On the other hand, an *informal structure* is the non-written structure built by individual participants or groups that shape ideas, expectations and agendas, and participants and groups bring to them distinct values, interests and abilities (Scott, 1992, p. 54). But, more important, the informal structure is always developed in a parallel manner to the formal one, but differs. It is transformed by asymmetrical power correlations in authority, subordination, insubordination, trust and distrust, among others, in a political manner. The informal structure is called *"how things are."* Every organization really operates by the combination of both formal and informal structures. For example, when the formal structure blocks operations, workers consciously still make decisions and work against the formally established procedures and policies to resolve that contradiction or paralysis.

Of course, the last formal-informal abstraction is an oversimplification of the complex organizational reality because the informal structure is developed at different levels: individual, group and whole organization, involving complex dynamics of economics, labour, power, social, political, affective and psychic relationships and interests that are interwoven among all members, including the owner and the manager. The formal-informal duality is a feature close to human, social, and organizational life, as it was exemplified by Goffman (1993, p. 31) in his work on social performance. This complex structure is manifested in the organizational culture that organization is, not has, as shared practices, beliefs, values, fears, myths,

phantasms. This complexity moves organizations toward the simple, mechanistic and predictable production system only operating for profit.

This formal-informal structure has implications for every organization, not only bureaucracies. The most evident implications for software development are important. Now, it is clear that the structure that should be modelled in software is the informal one, because it is the "*how things are*" structure that really operates, but in practice, the requirement analysts not use organization charts and manuals, among other documents (formal organization) as the primary sources for developing a software system in the organization. It was confirmed by IEEE Computer Society (2004, pp. 2-5) in the *Software Engineering Book of Knowledge* (SWEBOK):

"[the] software engineer needs to acquire, or have available, knowledge about the application domain. This enables them to infer tacit knowledge *that the stakeholders do not articulate, assess the trade-offs that will be necessary between conflicting requirements, and, sometimes, to act as a 'user' champion."* (IEEE Computer Society, 2004, pp. 2-5, emphasis mine)

This confirms that informal structure corresponds to the *tacit knowledge* or *expertise* of the members of the organization which software engineers are urged to assess during requirement elicitation. This knowledge acquisition and transfer as power is disputed by software engineers, software managers and business managers, and is one of the most important sources of conflict between management and software workers. Then, *software* may be conceived in an organizational approach as the partial model of the operative facet of the organization, as the "modern implementation of the rules, policies and business procedures, in a wide sense." (Zavala, 2004, p. 4)

Another implication of this formal-informal structure and dynamics is its universal occurrence. Therefore, is there any guarantee that a small software organization (supposedly formalized by any procedure) really adheres to it during day-to-day work? It is not possible because of cost and practical concerns, I think. Formalization requires an extra effort to maintain the closest relationship between formal and informal procedures, and consumes many of the SME's scarce resources (economic, time and human in nature). Therefore, formalization is the most practical organizing limit.

Formalization derives from the bureaucratic organization theorized in-depth by Max Weber (Miller, 1970, pp. 91-92). Weber found that rules and policies give certainty in behaviour by means of standardization (e.g., diagrams, workflows and organization charts) and behaviour regulation (working and making decisions following formal procedures). Bureaucracy is a rational model, a mechanistic model and a disciplinary system which is predictive, but provokes the so-called "vicious circles" when members use rules to defend themselves against the system and other members as well. In all organizations, members are "obligated," (informally) by their superiors or by situations, to violate formal rules to resolve organizational paralysis. This generates a *paradoxical management* but is tolerated by the hierarchy, always with a discretional and self-convenient basis. This universal violation of formal rules (procedures, policies, structures) is, to an extent, responsible for organization manuals always being outdated. Cornelius Castoriadis, for example, showed notably that "if workers strictly applied the methods and rules of the bureaucratic organization, it wouldn't operate one minute more" (free translation, cited by Enriquez, 1992, p. 129).

Another distinct approach to SMEs is developed by Rosa (2000).

THE ORGANIZATIONAL NATURE OF THE SOFTWARE PRODUCTION PROCESS

Approaches to the software-development process take two opposite postures: software work is routinized or it is not. The first position states that *software work is routinized* like any other traditional manufacturing work by successfully implementing neo-Taylorian principles in the software organization. This view is criticized for its political bias and lack of understanding of the nature of software work (Ilavarasan & Sharma, 2003, p. 1). However, in this approach, routine in software work is attempted by normative isomorphic mechanisms such as standard programming languages, programming, and analysis and design techniques (Unified Modelling Language, UML, for example), certification, training, learning, total quality management models (like Capability Mature Model, CMM and CMM-I, promoted by Software Engineering Institute), process standards (such as ISO 12207, ISO 15504 and others in progress by the Standard International Organisation), and quality standards (such as ISO 9001:2000). Specifically, quality models assume that quality is guaranteed by process and that it is objectively reflected in a quality product, but there are still many unsolved controversial conceptual and practical issues (Hackman & Wageman, 1995; Tuckman, 1994) more clearly in software organization. In spite of this, these models are used as prescriptive organizational models for universal adoption, attempting the standardization sought (Mutafelija & Stromberg, 2003).

This routinized view is called a *software engineering approach* that call for the application of "precise" engineering methodologies and techniques for software development in a "systematic" and "disciplined" manner. However, in day-to-day practice, there are many facts to the contrary. Moreover, not all organizations have adopted these normative models. Microsoft, a large organization, for example, does not adopt many of the structured software engineering practices commonly promoted by organizations like the SEI and ISO (Cusumano & Selby, 1997, p. 52) and many SMEs neither.

Tore Dybå (2001) studied software firms in Norway and concluded that rather than "'best practice' approaches such as the CMM, which rely on a predictable sequence of events," organizational members should favour an organizational culture oriented toward organizational learning based on improvisation, where technical procedures can prosper instead of imitating technical procedures (pp. 259-260). Dybå's conclusion is more important for smaller firms, I consider. In a similar way, the challenge in software organizations is to find ways to improve organizational skills that require firms to balance two seemingly contradictory ends: efficiency and flexibility. This is the conclusion of Aaen, *et al.* (1997, p. 431).

Regarding the implementation of quality systems, a study in Bangalore found that managerial control is not enhanced by quality certification procedures (Ilavarasan & Sharma, 2003, p. 6). Carmen Bueno (2000) used an anthropological approach to study a quality system implantation in a manufacturing plant in Mexico. She discovered that when getting a quality certification becomes a matter of a firm's survival, this goal becomes instrumental and an end-in-itself. Therefore, a certification process becomes a "great theatrical performance" for managers, workers, and auditors, in which everyone acts out his/her own role. In this performance, innovations and non-permitted procedures contravening the certification requirements are intentionally hidden from the auditor's scrutiny. On the other hand, numeric indexes are specified in such a way that they allow for manoeuvring and inventing them (pp. 43-46). As a result, well-filled-in forms, well-calculated numbers and indexes are available, but they become an end-in-themselves and no longer reflect the quality measure they intended to achieve, and the quality system has, in practice, destroyed itself.

The second approach contends that the nature of software work is a more creative, unroutinizable, complex, and intellectual activity than clerical work. In this approach, the software process is defined as mental-intensive labour such that the worker can solve the problem in many possible manners with enough leeway to use his/her creativity and imagination. Software workers clearly are knowledge workers that are not fully differentiated into conception or execution workers, and control over work is distributed among workers and managers not managers alone. In conclusion, software work seems to be unroutinizable at the moment and it will continue to be so for quite a long time (Ilavarasan & Sharma, 2003, pp. 1, 5, 6).

According to Dybå's (2001) conclusive study, it is clear that "organizational issues are as important in firm improvement as technology, if not more so" because "*the large literature on organizational theory, developed in non-software settings, has more relevance to software development than previously been recognized in software engineering*" (p. 269, emphasis added). Dybå provides insight into the most important issue underlying software production: how to improve it. It has yet to be fully addressed. My own hypothesis in this respect is that, in small business, adopting rigid and very quantitative procedures and management models is not recommended. Instead of those, more flexible and qualitative procedures and management models are recommended because they do not destroy the SMEs' competitive advantage centred on flexibility, improvisation, rapid response to the environment and efficient resource allocation, among others. This implies the necessity of a paradigmatic change of management from 'control' to 'that will facilitate learning' (Dybå, 2005, p. 420). Therefore, when adopting a quality system, it should not be an end-in-itself because it destroys itself, subsequently requiring a progressive, real effort to improve all organizational processes, that is, a practical formalization so as to be useful rather than normative, alone, because this is the real danger of some formalized (bureaucratized) initiatives, I consider.

Finally, the *"software paradox"* shows a contradiction in software organizations because, while the factory assembly line has been automated by "software engineers," ironically, they "have done a good job of automating everyone's work, but their own" (Mahoney, 2004, p. 15). This is relatively constant in many software organizations and it makes the software organization closer to a post-modern shop than a post-modern organization.

MAIN THRUST OF THE CHAPTER

Issues, Controversies, Problems

Management: A Management Paradigm in Crisis

During the sixties, the mass-production paradigm or the so-called *Fordist production system* was on a rise throughout the world. So it is understandable that, since the *NATO Software Engineering Conference* in Garmisch in 1968, Taylor's Scientific Management basic principles have been adopted as a management paradigm for the incipient software industry (Mahoney, 1990, p. 329). Since that time, it has been attractive to have an automated software factory with production lines and standardized and interchangeable software components, including interchangeable standardized workers as another production factor. For example, M.D. McIlroy's "mass-produced software" factory was considered the software factory on the other side of the Industrial Revolution (p. 331).

According to Frederick W. Taylor, the father of the Scientific Management the primary obligation of that was to determine the scientific basis of the task to be accomplished, based on four main duties:

- First. To develop a science for each element of a man's work, replacing the old rule-of-thumb method. The famous *one best way* acquired by management up to that point.
- Second. Scientifically select and then train, teach and develop the work force, instead of doing as in the past, when each one chose his own work and taught himself as best he could.
- Third. Heartily cooperate with the men so as to insure all the work being done in accordance with the principles of the science that have been developed.
- Fourth. There is an almost equal division of the work and the responsibility between the management and the workmen, the famous 'fifty-fifty' management rule. The *management takes over all work* for which they are better fitted than the workmen, while in the past almost all of the work and the greater part of the responsibility were thrown upon the men. (Mahoney, 2004, pp. 13, 14).

In addition, Taylor's system conceives monetary incentives only as a fixed and variable wage because "[h]igh wages give the workmen the spur needed to induce them to quit loafing on their jobs and to turn out more goods. The wage systems [...] usually consists of two parts: a base wage and a bonus. The base wage is customarily the same as the prevailing rate for the work in the community, while the bonus is a percentage of the base" (Keir, 1918, p. 528). Simplified, Scientific Management is the *one best way* of doing work. This implementation of this management system is called *Taylorism* and, ever since it came out, it has being criticized as ideologically biased, unscientific but scientific-discourse justified, extremely and mechanistic instrumented, not well-suited for universal application, and as a dehumanized production system. But despite this, Taylorism continues to fascinate and influence day-to-day management. Taylor's ideas persist as the roots of management theories and practice.

At present, it is clear that since sixties, style management has undergone its so-called fashions (Abrahamson, 1991, p. 255), fads (Clarke & Clegg, 1998, pp. 15-19) and "silver bullets" (Brooks, 1995, pp. 179-226). For example, Business Process Reengineering (BPR) and TQM had been become management fads (Miller & Hartwick, 2002, p. 26; Stewart, 2006, p. 84; Turner & Müller, 2003, p. 4) and did not fulfil their promise of providing a solution for predictive and successful management. In spite of failure in practice and criticism of the concepts underlying them (e.g., Hackman & Wageman, 1995; Tuckman, 1994), management fads still continue to be promoted, trying to prove their friendliness. Unfortunately, such miraculous solutions do not exist.

In the so called, humanist management, a different perspective than traditional management, Omar Aktouf promoted the idea that, in the ever-changing world of today, a new management type is needed which undertakes the challenge of utilizing the creative energy of all members of an organization, rather than focusing on the disciplinary aspects of behaviour on the job.

Survival in the future is to be able to mobilize, to the benefit of the organization, all intelligence, all brains and all energy, taking care of the environment to the maximum. It is to be able to achieve everyone working for himself and adopting an attitude of constant improvement, serenely, if possible, enthusiastically. And that can only be obtained by a proper climate, with job security, with an effective sharing of luck, good or bad. (Aktouf, 1998, p. 604, free translation)

An example of these new approaches distinct than Taylorist ones, appears in the work *Le management ludique: Jouer et travailler chez UbiSoft* [*Playful Management: Play and Work at UbiSoft*], discovered by Laurent Simon in a software organization in Montreal. He did not use a software engineering focus, but rather one of social sciences. Specifically, he used ethnomethodology

and participant observation to understand what he was observing and human sciences to interpret it. He made an in-depth observation and description of the software-game work process, studying managers and workers in action. Here are some of his illustrative conclusions:

The project is not a game, which goes without saying, but its structure, so similar to the structure of the game, can certainly welcome phenomena of a ludic type. (Simon, 2002, p. 387)

Finally, the will to create the capacity to auto-organize itself could be put effectively in parallel with the need to create and maintain a world environment "sufficiently good" that allows everyone to experiment, create and express themselves. (p. 392)

One of the most important consequences of this management approach is the possibility of fully developing creativity, and a conscious and voluntary commitment to work and a management conception:

The ludic logic of the légaliberté opens the way of a total, conscious and voluntary engagement in the double action of the possibility of a serene réflexivité on the value and the relevance of the action. The ludic conception of work also lends the activity a human value oft overlooked: one commits oneself because one wants to, because one can express itself and grow while learning, because one can hope and be oneself. (Simon, 2002, p. 397)

This ludic reading of the organization is not without consequence on management. It clears on a necessary recognition of the 'constructed' and 'situated' character of the organized activity, on a disclosure of its rules. If the game is product of the rule, to play the game well requires a good knowledge of the real rule, but especially the spirit of the game implies consenting to the game. [...]*The game to which one is obliged to play loses its character of game. Ludic liberty, supposes an engagement knowingly and voluntarily agree to, the sole guarantor of the motivation to play and to play well.* (p. 406)

In conclusion, this playful management approach transforms bureaucratic control into innovative and creative committed work, giving meaning to work and life-sense itself for workers and managers.

Lethbridge, Sim and Singer (2005), recently made a comprehensive compilation of techniques for studying software engineers, many of which were qualitative in nature. Similarly, a group of European researchers found evidence that main issues in software organizations are organizational in nature, not technical. A review of their research topics yields illustrative: improvisation (Dybå, 2000), empirical knowledge, evidence-based software engineering (Dybå, Kitchenham & Jorgensen, 2005), and culture and organizational aspects (Dybå, 2002, 2003). Sharp and Robinson (2004), for example, did *An Ethnographic Study of XP Practice* in the United Kingdom, attempting to be explicative. These authors use quantitative but predominantly qualitative observations based on knowledge management, learning organization and other organizational related theories. Much of this research is far removed from the dominant positivist software engineering approach which considers these "related disciplines" unacceptable because they are not quantitative. Simon's work clearly showed the power of social-science interpretive approaches and in-depth qualitative case studies as methodologies for studying software organizations and for management theorizing. They favour theorization because they are based on a set of richly detailed empirical observations. Unfortunately, often, both social-science and qualitative observations had been excluded by rational approaches, accusing them of being invalid and unscientific because of their distinct nature.

Software Project Management: A Social Constructivism Approach Required?

For software production, project management is the concrete exercise of management and it is the most difficult and problematic area of software engineering. It remits us to the infamous "software crisis" and to the project-failure phenomenon (Zavala, 2004, pp. 7-11). According to Harrison and Winch (cited by Hodgson, 2004, p. 85), the use of project management models and techniques has proliferated since they came into popularity in the late fifties following their use in high-profile technical endeavours such as the Manhattan Project and the Apollo space programmes. In this context, I consider that the adoption of software engineering as a project-oriented discipline at the Garmisch Conference in 1968 is no surprise.

Technically speaking, *project management* is "the application of knowledge, skills, tools and techniques to project activities to meet project requirements," accomplished through the application and integration of project-management processes of "initiating, planning, executing, monitoring and controlling, and closing" (Project Management Institute [PMI], 2004, p. 8). Also stated is that the project manager, in collaboration with the project team, is responsible for determining the appropriate rigor for each process and technique (p. 37). Project management promises frameworks and management techniques (Hodgson, 2004, p. 85) for *coping with technical and managerial challenges*, but most important, it offers *to predict and control people behaviour* by means of a strong bureaucratic control system (p. 88).

Project management draws upon the rhetoric of empowerment, autonomy and self-reliance, central to post-bureaucratic organizational discourse but, at the same time, immersed in a traditional bureaucratic essence producing much conflict. Project management is rooted in Taylorian Scientific Management and Fayolian administrative process, the hardest management approaches.

Turner and Müller proposed conceiving *project* as a *temporary organization*. They base this on the assumption that project pursues the organization's objectives and is managed by an opportunistic manager in conflict with the principal (Turner & Müller, 2003, p. 3). This instrumental approach is based on *agency theory* dating from the seventies, which is nowadays considered incomplete. Agency theory, rooted in the rational paradigm, is criticized because it is not be fully appropriate for addressing the complexity of organization. Of course, project management may be conceived as a temporary organization similar in complexity to every organization, with additional characteristics such as uncertainty, urgency, risk and highly dependent on the work team, manager and management. To this end, a more explicative and open organizational approach than agency theory is required. For example, I used organizational analysis in a socioeconomic approach to study the information technology department in a municipality in Mexico, with good results (Zavala, 2006).

"Software crisis," linked to software-project failure, has many justifications: (1) not carrying out a financial feasibility study, benefits being overestimated, and costs and time underestimated, frequently without solid bases for estimating (Haigh, 2001, pp. 79, 80, 89), (2) others (organizational causes) (Pinto & Mantel, 1990), (3) "software crisis" as a crisis of programming labour (Ensmenger & Aspray, 2000, p. 140), and (4) "software crisis" as a management crisis (Ensmenger, 2001a, p. 30).

Last new position that argues that "software crisis" is a management crisis is based on the fact that the "software crisis" solution most frequently recommended by managers is the elimination of rule-of-thumb methods (i.e., the old-time "black art of programming" or the "tacit knowledge" in present-day managerial discourse), among other principles of Scientific Management of an earlier era (Ensmenger, 2001a, p. 99). The author considers that real concern about "software cri-

sis" was the emergence of *the programmer as a new uncontrolled privileged knowledge worker.* Therefore, "*software crisis*" *is a management crisis* because of its inability to manage traditionally in a non-traditional industry.

"The apparent unwillingness of programmers to abandon the 'black art of programming' for the 'science' of software engineering was interpreted as a deliberate affront to managerial authority [...] The reinterpretation of the software crisis as a product of poor programming technique and insufficient managerial controls suggested that the software industry, like the more traditional manufacturing industries of the early twentieth century, was drastically in need of a managerial and technical overhaul." (Ensmenger, 2001a, p. 30)

The promised "scientific" managerial recipes of Scientific Management have become very attractive to software management because, as Dick H. Brandon pointed out, "the anarchic nature of programming meant that *management had to depend on the workers* to determine the pace of a project and that the insatiable market for programmers meant that *management had little control at all over the wage structure*" (cited by Mahoney, 1990, p. 333, emphasis added). The keyword in management has been *discipline* and the promise has been "to achieve managerial control over product, process and worker" (*idem*).

By now, the failure of project management to establish standards for selecting and training programmers is legendary. Despite Taylor insisting productivity was a 50/50 proportion, management played its role through selection, training, and supplying tools and materials relatively well. From the sixties to now, the science of management (whatever that might be) has not been able to supply what the science of computing (if there be such) had failed to establish so far, called a scientific basis of software production. (Mahoney, 1990, p. 333)

I also agree with Ensmenger that, despite more than four decades of managers having tried to settle down and rationalize software development as traditional manufacturing, developing software remains a distinctively craft-oriented and idiosyncratic discipline. Nevertheless, complaints about "the quality and reliability of software still plague software developers" and the "rhetoric of crisis continues to dominate discussions about the health and future of the industry" (Ensmenger, 2003, p. 174). In my opinion, software engineering is still in crisis but as a discipline, because of its management part.

Criticism of project management and of the management theory underlying it and its practice has occurred recently. Growing criticism of the intellectual and philosophical foundations of the discipline are rarely made explicit in project-management textbooks and publications (for example, Thayer (1997) is free of that) (Winter & Smith, 2006, p. 15), but are coming out. According to Mark Winter and Charles Smith, from the conclusions to *Rethinking Project Management EPSRC Network 2004-2006 Conference* was criticized: (1) that "conventional project management theory remains wedded to the epistemological and ontological foundations of the 1950s-1960s, with *its emphasis on machine-like conceptions of organizations and projects* (for example, Thayer, 1997), (2) that realistic assumptions about 'organisations' and 'projects' as entities existing 'out there' independently of the people involved" **(***idem***)**. For these authors, three paradigmatic changes were considered relevant at this conference:

1. *Projects as social processes,* not mechanical ones, toward "concepts and images which focus on social interaction among people, [...] the flux of events and human action, and the framing of projects... within an array of social agenda, practices, stakeholder relations, politics and power." (Winter & Smith, 2006, p. 5).

2. *A multidisciplinary approach*, not isolated, toward "concepts and approaches which facilitate: broader and ongoing conceptualization of projects as being multidisciplinary, having multiple purposes, not always predefined, but permeable, contestable and open to renegotiation throughout" *(idem).*
3. *A reflexive practitioner,* not a narrow one, toward "learning and development which facilitates the development of reflexive practitioners who can learn, operate and adapt effectively in complex project environments, through experience, intuition and the pragmatic application of theory in practice" *(idem).*

Surprisingly, previous recommendations by project management practitioners and theorists are oriented toward a reflective practitioner, only possible with broad knowledge and experience, as Aktouf (1998, p. 607) suggested, and far from the classic Master's in Business Administration (MBA) curriculum, as Stewart proposed (2006, pp. 86-87). The final criticism is against the rational paradigm underlying project management from a clear social constructivism or interpretive position, not explicitly assumed. This constitutes a change in discourse that facilitates a breakdown in its traditional position and implicitly proposes a paradigm change, in my opinion. This opens up project management to those theoretically, practically and ideologically "related disciplines" supporting a distinct discourse. *The most important consequence of that is the emergence of a necessary paradigmatic change in management, project management and software engineering*, I consider.

Now it is the moment to analyze software engineering as a discipline in crisis.

Software Engineering: A Discipline in Crisis Itself

Because of the "software crisis," it was argued that a new discipline for solving problems was needed when software engineering was invented in 1968:

In the late 1960s, in the wake of the 1968 NATO Conference, a new model for situating the professional programmer was invented. *Software engineering emerged as a compelling solution to the software crisis in part because it was flexible enough to appeal to a wide variety of computing practitioners.* (Ensmenger, 2001a, p. 30, emphasis added)

The term *software engineering* was indeed provocative, if only because it left all the crucial terms undefined (Mahoney, 2004, p. 9; Naur & Randell, 1969, p. 13). As conference organizers suggested, the solution to the "software crisis" was to convert the "black art" of programming (the programmer's tacit knowledge) into science, similar to what Scientific Management supposedly did, now with a worker's knowledge. Central to this "scientific" approach was programming activity. Software engineering's offer was very attractive because it "seemed to offer something to everyone: standards, quality, academic respectability, status and autonomy" (Ensmenger, 2001a, p. 31), besides the possibility of controlling the development process when the laws of causation were discovered. Finally, the hoped-for "silver bullet" was invented, but that promise has not come through:

[...] *for software developers to adopt 'the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering'. In the interest of efficient software manufacturing, the "black art" of programming had to make way for the "science" of software engineering.* (Ensmenger, 2003, p. 165)

Since the early days, software engineering was established as a goal to achieve (Ludewig, 1996, p. 25), but not as a legitimization and institutionalization of a genuine professional practice, because it lacked of widely accepted theory, practices and

professional culture, among others. This was demonstrated by the failed second and last NATO Software Engineering International Conference held in Rome in 1969. Some of that criticism of software engineering persists until today.

Results from the Garmisch Conference were determinant in both institutionalizing software engineering as a discipline during the seventies and in creating conditions for *a new stream of thinking* oriented toward *discovering those principles*. It is supposed that this new discipline should exist, but was never defined, nor posited as such, ever since it was born in 1968 until 1976. Since those early days, all these efforts have been oriented by managerial goals, but dressed in engineering discourse. In agreement with Nathan L. Ensmenger, a better approach to understanding software-engineering is to consider it a *movement,* "to better understand why it *succeeded* (*on a rhetorical level, at least, if not in actual practice*) where other systems and methodologies have failed miserably" (Ensmenger, 2001a, p. 30, emphasis added).

Thinking about the invention of a discipline as a series of interconnected social and political negotiations, rather than an isolated technical decision about the "one best way" to develop software components*, provides an essential link between internal developments in information technology and their larger social and historical context.* (Ensmenger, 2001a, p. 30, emphasis added)

A clear example of these political negotiations is the current "marriage and divorce" between computer science and software engineering as disciplines, as suggested by academicians and practitioners (Denning, 1998; Parnas, 1998).

Since the Garmisch Conference, both software engineering and management are looking for a miraculous solution that will finally resolve both the inherent complexity of software itself and of its production, and that will permit managing them. In Fred Brooks' words, they are looking for the "silver bullet" to "kill the monster."

Of all the monsters who fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, we seek bullets of silver that can magically lay them to rest. The familiar software project has something of this character (at least as seen by the non-technical manager), usually innocent and straightforward, but capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet, something to make software costs drop as rapidly as computer hardware costs do. (Brooks, 1995, pp. 180-181)

Software engineering emerged at Garmisch as the silver bullet for the "software crisis". But it is not a problem of shortcoming in computer science, technology or software innovation, but rather a managerial one. In 1986, Brooks stated that "… as we look to the horizon of a decade hence, we see no silver bullet" concluding that "*[t]here is no single development*, in either *technology* or *management technique*, which by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity" (Brooks, 1995, p. 181, emphasis added). This last sentence was valid for the decade between 1986 and 1995, as predicted, but, incredibly, it continues being valid for the two decades even to 2006. *Perhaps, it continues being valid because the management or social dimensions are the most complex issues in software production*, I would think. This confirms once again that the "software crisis" is one of management.

The official technical definition of *software engineering* states that it is "(1) The application of a *systematic*, *disciplined*, *quantifiable* approach to the development, operation, and maintenance of software; that is, the application of engineering

to software. (2) The study of approaches as in (1)" (IEEE Computer Society, 2004, p. 1-1; Institute of Electrical and Electronic Engineers, 1990, p. 67, emphasis added). Other definitions are similar. Barry Boehm proposed that formal definition of software engineering in 1976; and more recently, in his paper *A View of 20th and 21st Century Software Engineering in the ICSE'06 Conference* (Boehm, 2006), he upholds his same 30 year-old position. Since the 1968 NATO Conference, surprisingly, definition and discourse has not budged from its proclaimed (but not fully demonstrated) 'systematic', 'disciplined' and 'quantitative' intention, notwithstanding empirical evidences to the contrary (Lewerentz & Rust, 2000; Ludewig, 1996; Smith & McKee, 2001).

Although Bohem had said he himself is convinced that software engineering is and should continue being an engineering discipline, he considered it as "distinct" one, because its features definitively identify it as *a multidisciplinary discipline* which is oriented toward people. That "implies that the *relevant sciences* include the behavioural sciences, management sciences, and economics, as well as computer science" (Boehm, 2006, p. 12). Unfortunately, in practice, many of these "related" or "relevant" disciplines are not incorporated into software engineering, because many of them are not considered to be 'systematic', 'disciplined' and 'quantitative' as software engineering *pretends to be* and because they have an opposing epistemological and ontological discourse, definitively I consider. For example, in the Software Engineering Body of Knowledge (SWEBOK), in the chapter on Software Requirements (IEEE Computer Society, 2004, chapter 2), there is reference to them. They erroneously, continue being disciplines "related to" not "part of" software engineering. When they *are* incorporated, it is partially, only if they do not contravene the software engineering's central discourse. If they do, they are ignored. Finally, simply, I ask: why, are many of these "related disciplines" more social than technical in nature? Why would they have to be 'systematic', 'disciplined' and 'quantitative' if they deal with different study objects?

As previously shown, the "software crisis" was a rhetorical term used by proponents at the Garmisch Conference to interest the audience in the emerging software industry (Naur & Randell, 1969, p. 13). Since they emerged, data and statistics (and managerial discourse) have tried to support this catastrophic vision, but they are not very conclusive (Glass, 2006, p. 16). Today, it is wiser to consider the "software crisis" as an institutionalized myth (a common term used in neo-institutional analysis of organizations) subsuming the management crisis. This permits reinforcing both software engineering and management discourses urging the industrialization (in the manufacturing sense) of software production. However, software engineering is trapped in management (and software engineering) fads and fashions, too as management is.

Software engineering as discipline has being characterized by an unusual discursive stability, trying to consolidate itself since it emerged some forty years ago. This, from a Kuhnian perspective, may be interpreted as a paradigmatic hegemony trapped in its own paradigmatic myth: the "software crisis" and its 'systematic', 'disciplined' and 'quantitative' terms in its definition. So, it is pertinent to explore software engineering and its foundation so as to reinterpret them, because perhaps software engineering is considered improperly.

Next I shall analyze those facts associated with present-day software engineering.

- *Fact 1. The IT Paradox is a symptom of the failure of software engineering.* The so-called *Information Technology Paradox* (*IT Paradox*) has shown that organization productivity is not caused solely by information technology, as the data partially supports, except for large companies (Brynjolfsson, 1994; Brynjolfsson & Hitt, 2000, p. 45, 2003).

- *Fact 2. Software engineering as defined today has failed, because it has no clear theoretical roots.* First, we need a theory explaining software development more universally, not as exceptions valid only in particular firms. Second, software engineering and computer science are competing for ownership of the theoretical foundation, and this has led to an uncomfortable relationship between them. Third, perhaps it is time to recognize the pertinence of agreeing that so-called bodies of knowledge (BoK), such as Software Engineering Body of Knowledge (SWEBOK), are *ideal models* (in a Weberian sense) and they should be used as such, instead of being used for normative ends, as they are today. Fourth, radical criticism is emerging, questioning the project management epistemological and ontological foundations underlying software engineering (Winter & Smith, 2006, p. 5). This is the first open criticism of the project management paradigm from within the field in an international setting.
- *Fact 3. Disciplines related to software engineering are excluded.* In spite of the wide recognition of the need for a multidisciplinary approach to be successful in software engineering, in practice, many of those "related" or "relevant" disciplines are not incorporated, for example, they are not in the Software Requirements chapter in SWEBOK (IEEE Computer Society, 2004, chapter 2). Recently in Europe, a research current has emerged which is evidencing that classical software engineering approaches ones are not better for studying software development and software organizations. Tore Dybå is a good example of using a hard quantitative approach, combined with a qualitative one, in an organization learning approach (Dybå, 2001).
- *Fact 4. The institutionalization of software engineering does not mean a mature discipline.* SWEBOK justifies "unmistakable trends indicating an increasing level of maturity" with the increasing of software engineering education programs at universities, accreditation boards, professional societies, quality formal models and professional licensing, but this only confirms that an institutionalization process is underway, not disciplinary maturity. The application of a *systematic*, *disciplined*, *quantifiable* approach to the development, operation and maintenance of software is far from what practitioners really do around the world.
- *Fact 5. Software engineering is not engineering as such.* Fred Brooks is overwhelming when he proposes the essential nature of software (their conceptual, abstract construct) (Brooks, 1995, p. 199) and he declares that no software engineering or management technique would solve it. Neither client nor software analyst knows how to understand and tame the "beast" when organization becomes software. Both managers and software professionals are looking for the miraculous "silver bullet," but, unfortunately, all these initiatives have led to business fads marked for failure because the organizational nature of software in abstraction and production is not considered (Winter & Smith, 2006, p. 5). Software abstracts the operation in the organization, but it is so complex. Brooks (1995, p. 199) reveals a truth that many of us are very well aware of: clients do not know what they want, but software analysts do not comprehend the complex nature of the organization being modelled, neither do software requirements and software design, the most difficult key areas. From the perspective of software production process, there is broad consensus that it is not routine, but rather a creative process based on technical, business and managerial knowledge not easily formalized like some crafts that have become automated industrial processes.
- *Fact 6. Traditional management has failed to support software engineering.* Scientific

Management sounded so promising because of its supposed 'scientific' and 'disciplinary' characteristics for controlling "the anarchic and non disciplined nature" of programmers. Unfortunately, management has not been able to discover, as promised, a magic formula for achieving managerial control over product and development process. It does not exist, as Fred Brooks remarked twenty years ago. In a more critical sense, management is a myth (Stewart, 2006). Finally, some paradigmatic approaches appeared such as *management ludique* (Simon, 2002) and humanist management (Aktouf, 1998).

SOLUTIONS AND RECOMMENDATIONS

Software Engineering Requires a Paradigm Change

Considering the preceding, there is much evidence pointing to the fact that software engineering, as currently defined, is partially mistaken in its definition, tools and techniques and, far from being a type of engineering in the style of classical engineering. Ever since software engineering was invented, it has been unable to satisfactorily address its central objective: predict the software production process. The lack of professionalism in software engineering, as explored in-depth by Ensmenger (2001b), but it is not the problem, rather: "*software engineering has not progressed far beyond its roots*" (Mahoney, 1990, p. 335). Bound to its definition and perhaps the development of software engineering, it is not an engineering discipline, but a management one. This hypothesis is appropriate because software production is close to an organizational phenomenon and social in nature, and therefore, the theoretical and practical has not been fully addressed by the current software engineering approach. *Software engineering is an exhausted discipline* due to its reductionist conception, because the complexity of reality is huge compared to its narrow concepts, tools and techniques. It is time to reconsider the evidence emerging, which tries to support the wisdom of a more multidisciplinary approach, rather than its rigid 'systematic,' 'disciplined' and 'quantitative' one.

Software engineering needs a revolution (in the Kuhnian sense) that will achieve a transformation of the discipline itself and that develops a theoretical, practical, reality-based body of knowledge applicable to large software organizations and to small ones. We cannot continue looking for the "silver bullet," hoping that, some day in the future, miraculously, it will adapt reality to theory. We should embrace a different paradigm, conceived in short as those values, beliefs and techniques shared by the scientific community, in Kuhn words (Kuhn, 1971, chapter I), a new scientific theory explaining current reality should being created.

Experiments validating current theory are not possible because they are social experiments in nature and are "not fully controllable." But, each case, each organization is a live experiment itself and an opportunity for studying the concrete reality that will possibly permit building the respective explanatory theory. We should create a theory explaining current reality, not the one that we want or speculate should be. Therefore, we should be looking for what software engineering practitioners really do, just as Thomas S. Kuhn did when he proposed the decisive role of science history in formulating an adequate concept of science by looking at what scientists (software engineering theoreticians and practitioners, in our context) do, rather than what they say they do (*idem*).

I consider that an answer to that complexity, necessarily points to the roots of software engineering, to its definition, because perhaps it would be another kind of discipline, not necessarily engineering in nature and its current conceptual framework, methodologies and tools are inap-

propriate. If software engineering as a discipline does not open up to this discussion, it will possibly be replaced by those "related disciplines" that demonstrate understanding and solving the object of study better than a discipline that promised and failed to solve something. Such paradigm change is a necessity, not a whim. This change implies that the software engineering movement will restructure itself, based on changing its basic paradigms, scope, practices and institutionalization, as well as on a new discourse, practice and education.

Complementing and going beyond Kuhn, Gergen (1996, chapter 1), in a Social Constructionism approach, provides a useful analytic framework for conceiving paradigmatic change. His model of nucleus of intelligibility has three elements: (1) *theory,* conceived as the explicit version of the world, that is to say, "what is it?" (2) *method,* providing us tools to learn about this socially constructed world, and (3) *metatheory,* basic assumptions on the nature of the knowledge on which the whole model rests. The most important consequence of Gergen's framework is to consider that adopting a new paradigm necessarily requires abandoning the current one, looking for another discourse and using different arguments. Then, from outside, criticize and question the abandoned paradigm, make new formulations and promote the emergence and eventual adoption of the new paradigm. However, this transformation occurs only if those three elements of the nucleus of intelligibility (theory, methodology and metatheory) change. If they do not, you did not leave the initial paradigm behind.

Social constructionism is based on the following suppositions: (1) terms used to define the world are not provided by the object itself, neither are they part of things, but are the product of agreements or naming them, privileging certain interpretations and not others (Gergen, 1996, p. 72); (2) terms used to describe the world and ourselves are the result of the social interaction of people living in historically and culturally located communities (p. 73); (3) the degree to which a way of naming the world remains over the time does not depend on its empirical capacities or its empirical strength, but on the vicissitudes of the social and historical process (p. 75); (4) concepts do not have meaning in and of themselves, their meanings are derived from language games and their associated rules (p. 76); (5) the unique way in which we understand the world is to acquire another conceptual framework because it is sole possibility of seeing something different from what we are accustomed to seeing, understanding and naming in that new reference system (p. 78).

Therefore, I propose that paradigm change in software engineering could be made using one of four alternatives:

1. *Giving more emphasis to "related disciplines"* as "part-of" its own body of knowledge, (including organization studies, as a central one), being "tolerant" of discursive contradictions, because their not 'systematic', 'disciplined' and 'quantitative' definition and contradictions with. If not, it will prolong the current state of crisis without resolving it. If software engineering, as a discipline does not open up to this discussion, it may be replaced by those "related disciplines" that prove to understand and solve the object of study better than software engineering promised and failed.
2. *Splitting its present range* into two complimentary, well-known disciplines. The first is *software engineering*, but redefined as a technical discipline, focusing on the technical aspects of software programming and replacing current 'systematic', 'disciplined' and 'quantitative' features with broader "scientific" far beyond any positivist or post-positivist classic approach. The second discipline might be *software management* or *software sociology*, focusing on the management, organizational and social aspects of software production conceived as a multidisciplinary and multiparadigmatic field. It focuses on the

software production process, embracing all necessary, present-day "related disciplines" such as management, linguistics, anthropology, sociology, psychology, psychoanalysis, labour studies, cultural studies, organization studies, and science and technology studies, among others. With this division, software analysis and design could be redefined into two separate, but complimentary, areas: one defined by business requirements and another by software modelling.

3. *Creating a new discipline: software sociology*. If software engineering does not choose either of the last two options, then the unique solution is to create a new discipline that focuses on studying software as social fact. This discipline, necessarily multidisciplinary and multiparadigmatic in approach, would address the implied organizational complexity. The discipline would exclude technical aspects of software production, approaching it in a social, organizational and management manner, incorporating whatever disciplines it needs for an adequate study and understanding of that so complex object of study. Quantitative and qualitative approaches are accepted as complimentary support.

My conclusion is clear. Software engineering is in crisis principally because the object of study is too complex to be approached with the current narrow mechanical conception of organizations, software production and software itself. Software engineering is trapped by its own definition (its management component and its disagreement with computer science) and by its myths ("software crisis" as management crisis and the "software factory" as a non traditional factory but managed using a traditional approach), but it is fundamentally trapped by its own underlying paradigm: Scientific Management. Presently, it is clearly accepted that such a vision is insufficient. Software engineering as a discipline has succumbed to managerial whim and to the accelerated inertia of business, based on a financial logic of the short term, competition and economic war making economic and social progress unsustainable. I am not against management, rather for its transformation toward something that allows profit, but at same time permits development of the organization, employees and society in which it operates. In other words, management should turn into administration and serve humanity, not the other way around.

FRAMEWORK FOR SOFTWARE ORGANIZATIONAL ANALYSIS

Using Gergen's (1996) analytic scheme, I assumed the pertinence of using social constructionism as the metatheory, permitting to state basic assumptions for the conception of the world and the way of approaching it. The ontology of organization used in this framework is organization as action, as a verb, as a process of organizing, as a discourse, as Clegg and Hardy (1996, p. 3) suggested. I use an organizational change epistemology using process studies as narratives of actions and activities by which collective endeavours unfold, as Van de Ven and Poole (2005, p. 1387) and Tsoukas and Hatch (2001, pp. 984-985, 996-1007) suggested. In this sense, narratives are our main resource for studying organization.

Organizational analysis should be conducted without any prior hypothesis to be tested, but the object of study (the software organization) should be constructed progressively during social interaction between organization members and researcher. Therefore, progressive and cumulative research is pertinent, in an attempt to discover regularities, contradictions, events, sequences and actors by opposing all points of view, from owner, administrative workers to base workers.

In practice, *ethnomethodology* and *participant observation* would be used when possible, trying to observe and describe organization as detailed as possible. At same time, superficial and in-

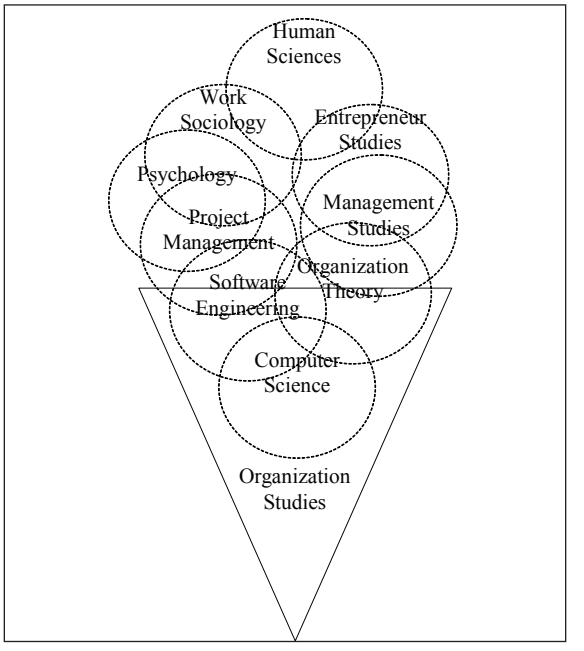
depth interviews of representative individuals in a semi-structured approach, oriented toward the subjects in question around in a *life-story* approach (Bertaux & Singly, 1997, pp. 31-50), are appropriate. In general, interviews explore meanings involving personal, academic and labour trajectories, engagement and identity as a member of a company, and as participant in work activities and projects; immediate and midterm expectations regarding work and profession; the opinions of bosses and owners regarding the problems and solutions proposed, reasons and causes why someone is leaving the company. We should encourage individuals to narrate each topic, looking for places, times, events, sequences, actors and situations as precisely as possible. Therefore, we articulate these distinct individual narratives or life-stories, each explaining its own point of view or position, but the stories are interwoven in agreements and disagreements. A sequential reconstruction of organization history, looking for events that shaped the present-day organization, is suggested.

Accordingly, I had proposed three conceptual tools for analyzing software organizations in an organization studies approach. The *first conceptual tool* is a *typology for classifying software organizations*. Jorge Carrillo and Arturo Lara, as well as Arturo Hualde, proposed that *maquiladoras* (industrial in-bond companies near Mexico's northern border) evolve through a generational typology based on Weberian ideal types. Each category has been defined as a "generation" from first to fourth. They considered the next generation to be more competitive than the previous one (Carrillo & Lara, 2005, pp. 7-12; Hualde, 2003). Carrillo and Lara considered an ideal work-evolution process to be as follows: *Manual Work → Rational Work → Creative Work → Not Material Work - Coordination Work* (Carrillo & Lara, 2005, p. 29). Similar to *maquiladoras*, I consider that software organizations may develop these, but there is no research on this yet. Software organizations support industrial firms that are adopting information technology for business operations. There is even a large Mexican software company that identifies itself as a *software maquiladora* and international advocates conceive software development as *maquila* (i.e., manufacturing) (Hualde, 2004, p. 19; Sinclair, 2003) because it involves the off-shore software business.

Considering Carrillo and Lara's typology, software organizations could not be fully classified into any of the four *maquiladora* categories. However, because the type of work (from a managerial point of view) and labour conditions, it would be classified as third or fourth generation. The software-organization typology, inspired in maquiladora-generation typology, considers only a few organizational variables based on theory and practice (Table 2). It is only a descriptive tool because only an in-depth organizational analysis would give us greater insight into understanding its internal structure and dynamics.

The *second conceptual tool* uses the analogy of the ice-cream cone to simplify understanding

Figure 1. Metaphoric organization studies as a multidisciplinary approach



(Source: Own elaboration)

Table 2. Typology for classifying software-organization generations

Dimension	1st Generation Software Firm	2nd Generation Software Firm	3rd Generation Software Firm	4th Generation Software Firm
Software Engineering Focus	Programming	Testing	Analysis & design + Project Management	Project portfolio
Market	Local	Regional	National	Global
Rewarding	Material	Training	Career	Life
Organizational Paradigm	Rational	Integration	Knowledge	Humanist
Management Focus	Product	Process	People	Culture
Management	Command	Technology, Project	Creative Process	Professional & New Approaches
Main Analysis Level	Individual	Group	Project	Organization
Knowledge	Basic Individual	Techniques, Process and Methods	Project Methodologies	Multi-disciplinary Research
IT Infrastructure	Incipient or Manual	Basic Planning	Encourage Design (Creativity)	Knowledge Management (Industrial)
Annual Turnover ($USD)	0.25 thousand	1 million	50 million	> 50 million
Human Resources	Cheap & New	Consultants	Highly Skilled Experts	Highly Skilled Researchers & Innovators
Full-Time Employees (equivalent)	< 50	50 – 99	100 – 500	> 500
Flexibility on Labour	Little	Functional	Projects	Workgroups
Firm Age (years)	1-5	5-10	10-20	> 20
Entrepreneurship	Venture	Survivor	Consolidated	Growing
Main Competence	Software Programming	Project Management	Software Design	Knowledge Production & Management
Business Focus	Relationships	Marketing	Invitation	Competence
Work	Manual & Craft	Rational & Specialized	Creative & Expert	Non-material, Symbolic & Professional
Sub teams/Projects Number	1 – 5	6 – 10	11 – 50	> 50
Managerial Control	Supervision	Delegation	Coordination	Self- Organization
Labour	Freelance, Halftime, Precarious	Fulltime with Social Security	+ Limited Stock Participation	+ Partnership, Decent Work
Formal Degrees	Studying and Bachelor's	+ Bachelor's in-hand	+ Specialization & Master's Degree	+ Doctorate
Experience (Full-time years)	< 2	2 – 5	5 – 10	> 10
Apprentice	Novice	Apprentice	Mentor, Consultant	Expert
Stock Ownership	Single	Family	Partnership	Social
Organizational- Size Category	Micro & Small	Small	Medium	Large

(Source: Own elaboration)

the multidisciplinary approach of organization studies. The wafer represents organization studies and the scoops of ice cream the disciplines participating in a dialogue regarding the object of study, each contributing its own flavour (paradigm, discourse, theory and techniques), looking for coincidences and divergences that allow achieving a more complete understanding of the object. (See Figure 1) Disciplines can be added as needed.

The *third conceptual tool* is a framework for the *organizational analysis* of software organizations, inspired in the idea that there are many levels of analysis similar to Enriquez (1992). This framework is conceived as an ideal type, not existing in the real world. I propose five levels of analysis (psychic/symbolic, individual, group, organization and inter-organization), three ideal actors or roles (engineer, project manager and owner) and three disciplines (software engineering, project management and entrepreneur theory) for addressing software organizations analysis (See Table 3).

In Table 3, each column corresponds to a principal role: *engineer* (skilled workers: brand-new or student trainees, junior and senior consultants, experts, non-technical workers and administrative staff), *managers* (project manager, program manager, technical leader) and *owners* (partners, top management, stockholders, managers, proprietors). In tune with these actor roles, a main discipline provides discourses, theories, methodologies, practices, techniques and tools for addressing its concerns. From left-to-right

Table 3. Framework for the organizational analysis of software organizations

Analysis Level	Ideal Organizational Actor		
	Engineer	Manager	Owner
Inter-Organization (Network)	Job Prospecting Career Development (Mercenary Work)	Political Relationships	Entrepreneur Networks, Strategic Alliances (Business Relationships)
	Competence	*Support*	*Trust*
Organization	Engagement & Commitment (Organizational Knowledge)	Technical & Managerial Leadership (Project Management)	Organizational Culture, Leadership & Management (Social Rewarding)
	Rewarding	*Leadership*	*Power*
Group / Team / Project	Work Group, Social Relationships (Membership & Mentoring)	Personnel Management, Work Organization (Leadership)	Discourse, Ceremony & Goal
	Task	*Goal*	*Business*
Individual	Learning & Knowledge (Expertise)	Rewarding & Group Relationships	Business Management
	Software	*Project*	*Enterprise*
Psychic / Symbolic	Values, Beliefs, Fears, Myths (Psychic Commitment)	Work, Challenge & Responsibility	Entrepreneurship, Owner Personality & Family Support
Discipline	Software Engineering (Software Development)	Project Management (Software Management)	Entrepreneur Theory (Venturing/ Entrepreneurship)
Organization Studies (OS) (Other disciplines needed, as part of OS)			

(Source: Own elaboration)

and from bottom-to-top, cumulative complexity is taken into consideration, not as a linear or causal pattern, but rather as a continuous, complex one. For example, *Project Manager* at *Individual Level* is surrounded by eight quadrants. Then its analysis may include both: the three *Software Engineer* (left quadrants) as well as the three *Project Manager* (right quadrants) and both upper and lower quadrants. Obviously, the topics in each quadrant are illustrative, not unique, and one should include others as needed. Between each level of analysis is a transitional object (real or conceptual) that acts as an attractor between contiguous levels and materializes the analysis. For example, Software permits materializing the internal creative energy (the so-called Freudian *pulsion* of the individual aspects: learning, knowledge, expertise) and its psyche (values, beliefs, fears, psychic commitment). Only the symbolic level permeates all the other levels, manifested in each individual as culture, in this case, as the organizational culture that software organization is, not has. At once, there expose briefly the obtained results of applying the framework in the organizational analysis of a small software organization.

SMALL SOFTWARE ORGANIZATIONAL ANALYSIS: A CASE STUDY

The case study analyzed is a small Mexican software organization. It was studied, carrying out an organizational diagnosis oriented toward evaluating the adoption of a CMMI quality model. The software company is located in Mexico City, has fewer than 50 employees and it develops software for large private firms. The target of the organizational analysis was that it was allowing define the strategies towards the implantation of the quality model CMM. Only the main findings using the organizational analysis framework are presented at each level of analysis. I quoted representative or illustrative words, phrases and concepts obtained from the interviewed people.

Inter-organization level. At this level, I discovered two main actors: the owner and the project manager. Both people use their relationships with the managers of client firms for project concourses, contracting and development, as for prospects for other business deals. The inclusion of the client makes labour and production relationships more complex. The small software firm's business model clearly was based on high project-profitability because it contracts non-expert people who are outsourced as experts. The client firm and the provider (the software firm) are aware of this, but both client and provider "cloak them in mutual complicity": the client turns a blind eye and provider does "extra work" outside the scope of the contract. This is a power game, as "blackmail hoping for another project contract." The owner has a broad relationship network with business partners, friends, ex-workmates and others. This contact-network data is updated monthly through phone calls, business breakfasts, lunches or dinners, attending tech events, as well as other social events. These meetings are formal or informal in nature. In conclusion, at this level, political and social relationships are vital to doing business.

Organizational level. At this level, many elements were discovered. Using the "interiorized charts," as I named them, I discovered as many organization charts as there are members in the organization. Each one reflects the structure of authority or control which each individual believe exists (or perceives). Only the one drawn by the owner reflects the "official chart." Using this technique, it was possible to analyze the "real" organizational operation (informal structure), specifying the area to which the interviewee belongs. Later, this and other paper-based techniques were used to interpret the software-project dynamics, based on concrete examples, not idealized ones.

An organization incongruity was discovered, manifested in two completely divergent dis-

courses: public and private. The public discourse depicts a great company: the best, the winner, a unique one with the best technology and best employees, the one that has done major projects (and to demonstrate such, exhibits on its website the logos of companies that were clients at some time, be they good or bad results, but already part of the firm's past and current curriculum). The size, image and discourse frequently contrast with reality. On the other hand, the internal discourse directed to employees for the purpose of producing a common image about the firm is not contrary to the inner reality of the day-to-day labour reality. For example, in practice, workers are not as important as the public discourse says and incentives are not paid out as promised, among others inconsistencies.

The owner of the business, himself defined as "partner-president," exercises an "authoritarian" management style with little delegation to his immediate subordinates (managers) and a prize/penalization system against wages to "indemnify" the firm in cases of "unacceptable acts" or "non-compliance with commitments" by employees. Notwithstanding this punitive control style, it frequently does not obtain the desired behaviour by employees. This is widely sensed by the owner, but he considers it "appropriate." Owner argued that his punitive (carrot-and-stick) management style was "humanist" and that it was learned it when he studied his MBA in a main business school in Mexico City some years ago. Some employees resist managerial control, allowing the penalization, but not changing to the behaviour desired by management. The owner is undergoing a leadership crisis and personal discrediting, though he is considered as a "good technician." None of the immediate subordinates trusts him. One of them demanded he be "directed by his boss." Engineers use the "cycle of terror" (letter of resignation) as a weapon for negotiating a pay raise with owner, believing that they put him into a corner. Employees do not feel comfortable using it, but it works, they argued.

The reward system is solely "money-centred," based on a *homo economicus* conception that supposes that money is the main stimulus for employees. However, they look for "friendship, companionship, solidarity, love, recognition," among other rewards in the organization, because "money by itself is cold." This demonstrates the non-validity of this punitive style. This reward system is based on a fixed, plus variable-wage, calculation. The variable rate is much more important and it produces anxiety and behaviour oriented toward formally satisfying the "quality-assessment form" and the client's project manager, because he fills out the form as an act of power, rather than one of real quality control, then "quality is anything he wants."

Managers (project managers, technical managers, department heads or managers) are the technicians with the most time in the company and they have been promoted to their positions by the owner. Two of them have no formal (academic or business) education for the position they hold. One manager with management training is considered generally grateful and successful by his subordinates and client managers; the other two, not so much. Frequently managers' authority is harmed by the director giving contrary orders. Coordination, delegation and leadership, as well as education, are the main problems faced.

Group/Team/Project level. At this level, I have focused on the production system. The engineer works at the client's office. Sometimes it is an office with furniture, computers and network services for "consultants" (software programmers or software developers), "project leaders" and "technical leaders" (rarely same person). This produces tension within the work team. Other times, the client conditions large rooms or surplus facilities with a minimum of investment. Tables, chairs, electricity and communication cables are in disarray and aesthetically ugly. There are bad working conditions. It is hot or cold, according to the season, and no climate control. Rooms have no ventilation and no windows. Sometimes there

are cooling apparatuses, but they do not circulate fresh air. Other times, the development team works in meeting rooms fitted as work places.

People are physically lined up in symbolic production lines, frequently made up of several work teams belonging to different suppliers and all operating physically and emotionally apart from the other workmates, even though they work together in the same place. Consultants, as they name them-self, do not have Internet access for peer-to-peer communication (chat), e-mail or web browsing. When searching for technical databases, tips, files and other information resources, workers use Internet cafés or other Internet facilities.

Work organization is based on "its own private methodology," says the public discourse. In reality, it does not exist. Project management is one of the organization's main problems. There is ambiguity in control, coordination and authority, because of the conflicting roles between project and technical leaders, among others. Frequently, a labour conflict becomes a personal one. Project and technical leaders compete for leadership and control over the work team and project. Frequently, the "leader-leader" (the "real leader") recognized by work team is the technical leader instead the project leader. Although there is a weekly report, what is portrayed in such reports frequently does not correspond to reality, but managers say they have projects "controlled" (through the reports), even though they are really behind. Project cost is "controlled" by penalizing the variable wages of managers and consultants, "against them" as employees say.

In day-to-day work, consultants do not use commonly known software-system-development methodologies. Rather, each ones does their work as "they can" and "they achieve goals," using their "own methodology" (whatever it be).

Individual level. At this level, a labour cycle was found. Employment in the organization is competitive, but recruiting process is quite expensive; so that recommendation by organization members is preferred. It produces better performance and forms a "sense of group" when it is finally set up. Unmarried young candidates and those "hungry for learning" are looking to be recruited as trainees. Thereafter, managers offer the next positions progressively: "*associate consultant*", "*junior specialist consultant*" or "*senior specialist consultant*". The "*junior manager*" and "*senior manager*" positions are exclusive those "privileged" by the owner. This career path occurs in a progressive engagement-conflict game cycle, called "cycle of terror" by employees. It consists of being involved in a project, becoming a critical "resource" in day-to-day work, threatening resignation, negotiating a pay increase, and getting a new position or a pay increase, or leaving the company if the strategy fails.

This career path is not conceived as for the real progress of employees, but rather as a means for intensifying work according to high profits and short-term business logic. The business model is clear, because it is based on hiring non-experts for "sale" (outsourcing) as experts. The entrance wage is $450 U.S. (2005), with maximum hikes of 20% per year, assigned discretionally in the power game before described. This wage has always been the crux of the matter ever since the company started. It is "planned" that a worker will stay on nearly six years to get to earn $1,650 U.S. (a relative decent wage) as senior specialist consultant, if not fired earlier. During this period, the worker has already "attempted" leaving the company several times, always using the "cycle of terror." Around the sixth year working at the company, a consultant begins to be less dependent on his parents because he lives alone or forms his own family, and income growth becomes critical. Perhaps he will leave the company. If not, he may fall victim to economic pressures, computer illnesses (neurosis, colitis, gastritis, among the more serious ones), and have a shattered family life because of work. For six years now, he has being working "without a labour contract" or "paid by the job" as a freelancer (but working for a single employer). Perhaps, he may now realize

he has no any benefits, such as getting a loan or applying for a housing loan. Perhaps he realizes he has lost a lot by having a job without benefits. If a worker leaves the company, there is more uncertainty and precariousness outside than inside. Now he has the experience, but he looks forward to from three to six months of temporary jobs and low wages. He does not really have a competitive professional career.

Psychic/Symbolic Level. At this level of analysis, I concentrated on the inner aspects that individuals express about themselves, the organization, work and co-workers. It has been discovered that, once an individual knows the mechanics allowing him to obtain positions, wage differentiation and discretionary assignment of positions and wages, he becomes "disenchanted with the company" and begins to look for alternatives. At that moment, a wage-negotiation process begins that will continue on for a long time. Individuals are aware of attempts at manipulation by managers from the first or second time promises are not kept. Therefore, trust in the system and in individuals is lost and both enter into a game of power and mutual convenience until such time as someone does not accept something and the relationship breaks down.

I found little identification with and commitment to the company. It is for being trained, but not for making a career: "it is not for making a career it is for three or four years. It doesn't last any longer" or "it doesn't even pay anything." The company's values and the individual's do not coincide completely. No company values are interiorized by workers. Those come from their families.

Identification with the company as a common project does not occur. The company only offers money, though it is not enough and does not provide any human warmth: "money is cold." The company and its worker are separate entities: "I work for myself" [not for the company], then there isn't a serious courtship. Employees do not "fall in love" with the company.

Employee commitment is achieved when an individual is appreciated as a member of a select group (the company), it trusts him, assigns him responsibilities, he transfers his commitment over to the client, and the commitment is achieved when the company represents the ideal the person is looking for.

This analysis of a small software organization uncovers aspects that should be resolved at each level: non-quantifiable and impossible to acquire from a software engineering approach. In this case study, I carried out semi-structured interview using, as attractors, the concepts expressed in Table 3 between levels of analysis. As result of this organizational analysis, problematic technical aspects were limited to a lack of practical application of programming methodologies and to a need of technical mastery of the technologies used. In the end, they succumb to the pressure of time and the client. On the other hand, management-related aspects turned out to be more important, permeating all activities in the organization. The dynamics of the organization are tinged by management aspects.

The apparent urgency to achieve CMM-I is really the owner's desire, based on the assumption that once certification is achieved, the company will be more attractive on the market. Another assumption by the owner is that the production process will become predictable and independent of workers, because "process drives work." The promise of a quality model does not always give results. After the owner has made various attempts over several years, it has not been possible to implement even small effective control policies and practices, because they are discretionary in nature. This confirms that there is a real practical limit in the level of formalization because of the high consumption of resources. Managerial control does not control anything in day-to-day, except wages, because weekly reports do not substitute control. It is an illusion.

In this organization, there are many managerial contradictions because of management style.

However, the owner's immediate past reflects a different work style, similar to mentorship. Although management should generate favourable conditions, liberating the creative energy of all the members of the organization and channelling it toward work, it is not occurring in this company. This repressive management style has already become institutionalized. Managers, project mangers and technical managers practice it, occasionally magnified by a possibly disproportionate exercise of power when they do not get the desired behaviour from subordinates. This rational or rather punitive management system, inspired in classic American management, produces individuals who are never really committed to the organization, neither to the managers or owner. For example, workers with more than 5 years of experience in the company are not convinced of staying and making it a career because the business model pushes them out. If management substitutes them for other cheaper workers, they would stay. After both monetary and emotional acts cancel each other out between manager and worker, the balance is zero. There is no gift, debt, gratitude or received favours that have to be given in the future, therefore there is no commitment. In addition, the company promotes values that are contradictory in practice, so that the labour relationship transcends the economic value, and the lack of emotional aspects employee-employer are transferred to the client. Since the consultant and client are involved in a recognition-commitment game, the boss only represents the role of paymaster and wage negotiator. This demonstrates that the *Principle of Gift*, proposed by the social anthropologist Marcel Mauss at the beginning of the twentieth century, and expressed by means of three obligations: to give, to receive and to return (goods, obligations, symbols, etc.) (Mauss, 2002[1924]) has full validity in this type of organizations. I conclude that the symbolic aspects come out the monetary ones and that "the substitution of a rational economic system for a system in which exchange of goods was not a mechanical but moral transaction, bringing about and maintaining human, personal relationships between individuals and groups" (Mauss, 1970, p. ix).

The personnel manifest a big desire to achieve balance and justice within the company. They say they suffer anxiety to a greater or lesser degree. Management does not listen to employees and acts against itself by destroying the trust between both.

This analysis confirms that many aspects, except software development methodologies, are management in nature: labour and management discourses, labour relationships, power games, incentives, leadership, reward system, management style, work conditions, project management, career management, involvement and commitment, trust, symbolic aspects, among others. Therefore, organizational analysis is appropriate.

The first conclusion is that, under these particular conditions, any attempt to implement quality models is risky. While the company is trying to achieve certification, the personnel will stay to learn, but, once granted, they will look for better job offers outside the organization. They do not have a future (as they have seen from other co-workers fired) or they freelance because they earn better wages but are less committed. If a "software engineer" decides to freelance after a frustrated round of wage negotiations and this frustration is repeated, he will behave as a *mercenary worker,* always looking for better paid jobs instead of working more committed. This learning experience is more significant for younger workers because they have seen a "bad example" of work cooperation, a competitive work "organizational culture" instead of a cooperative one. Possibly, when they leave the company, they will believe all software organizations are the same, thereby producing, non-cooperative power-game behaviour.

I have found this framework useful in studying organizations. Six successful diagnoses of the organization of small software organizations using this framework confirm this (one described in Zavala, 2006). This framework has evolved since first conception and may still evolve.

In my consulting practice, I have found that every organization should be considered distinct. There are no universal recipes, at least in this practice.

FUTURE TRENDS

Disciplines are opening up their own paradigms toward a more inclusive and more multidisciplinary approach. For example, medicine today accepts and includes homeopathic medicine and alternative, traditional, sacred or natural medicine in official medicine, as occurs in Mexico recently. This shows a paradigm shift. These epistemological and ontological issues are major concerns that research should address. However, it still has not happened in the so-called "hard," "rational" or "empirical" disciplines such as engineering, because they are tied up in the most pure objectivism and reject contributions made by "soft" sciences such as the social sciences and humanities when they contradict their "hard" discursive positions. Surprisingly, it is beginning to be accepted in engineering disciplines that the positivist approach is not the only one valid and that it has serious constraints. Software engineering is an example of a discipline that has not been able to consolidate itself because of its roots. More research is needed on the ontology and epistemology of this discipline. Law is another case of a hard discipline, where we can see how a scenario is frequently interpreted different ways, and the verdict is the result of political voting, cloaked in legal language. Where is the so-called objective action? Language is so important and linguistics is another discipline of concern and of promise to software-engineering research.

Research in science and technology is needed to uncover (not discover) how software development really occurs. Research methodology based on extremely restrictive rationalism is unable to address that complexity which is not measurable or quantifiable. I believe this will provide space for better qualitative appraisals and case studies or for using another qualitative methodology. Qualitative research, with no previous hypotheses to prove and rich in detailed observations, has been able to demonstrate the falsehood of stereotypes or beliefs, even building a new theory. Henry Mintzberg and Laurent Simon are examples. Therefore, one of the most promising fields for research is *action research*. The contributions of human sciences as disciplines are slowly being discovered as the "other" disciplines, for example, in management, the emergence of concerns of ethics and social responsibility in corporations.

Given the criticism arising among practitioners and academic groups, mainly in management and project management, it is possible that, in the short term, research on software projects, software engineering and management will multiply and, as a result, organizational issues will gain in importance.

Another promising field of research is that allowing us to create specific theories, techniques and methodologies. Research done in Australia and Europe demonstrates this. Latin America is a virgin field because of the specific conditions for developing particular software organizations.

CONCLUSION

The main conclusion is that software organization is not only a function of profit, but a complex social system, far from a simple rational system that can be analyzed from a powerful perspective, as is illustrated by the organization paradigms cited. However, this requires a multidisciplinary outlook such as organization studies. Software organization is a paradoxical post-modern organization, because of the nature of software, production and economics, but is closer to a post-modern workshop, because of its production system.

Generally speaking, small organizations and small software organizations are particularly complex because its challenge is to compete

competing with large companies. Conditions are favourable, with the ability to innovate and the capacity to make quick changes, but, most important, their organizational formalizing limit is their most important constraint to an attempt at implementing any rational managerial system. Therefore, formalizing should be the result of an organizing process, instead of a prescriptive end in and of itself. Discovering the equilibrium point specific to each organization is the most important challenge for any discipline attempting to do so.

Recognition of the specific formal-informal structure and the organization dynamics of each organization is an issue of major importance because of the profound implications they have on organization-formalizing dynamics and its practical limits, software abstraction and software-engineering theory and practice.

Software production (development and maintenance) is a social process or, more precisely, organizational in nature. It includes a technical facet and has profound implications for software-engineering education requiring reforms to education curricula.

The rational management paradigm has limited application in present-day software organizations because they are different in nature: people and production process are not like industrial manufacturing, where even this paradigm has limited success. New humanistic-oriented management styles are apparently more appropriate for these knowledge organizations and their scarce, highly skilled knowledge workers. It is clear that project management, as a management discipline, requires a paradigmatic shift too, oriented toward more humanistic concerns.

Using a science and technology studies approach, I have demonstrated that software engineering is an invented discipline, built on the "software crisis" rhetoric. Limited real advances in software engineering as a discipline are clear. It has only been successful in a discursive way, but, in methodologies, techniques, theory and practice, it has failed miserably. Software engineering has trapped itself, because it attempts to solve a concern that is not technical in nature, as has been demonstrated. This obliges us to conceive software engineering as organizational in nature, with profound implications. Therefore, software engineering should change its current paradigm, accepting those "related disciplines" as "part of" itself. It should be reshaped, redefined, excluding its supposed management scope, so as to guarantee its so-called "quantitative, disciplined and systematic" properties, or succumb to those successful disciplines.

The framework proposed has resulted in a powerful conceptual tool for analyzing software organizations. But alas, for you to use it and interpret the distinct facts, histories and narratives, a paradigm change in you yourself is needed. From my own experience, this was a prerequisite. I abandoned my rigid thought as an engineer when I immersed myself into organization studies as a new and strange discipline. It has being the most complicated intellectual task I have attempted. Only after that, was it possible to reconsider my prior engineering knowledge and reincorporate it as another discipline, not as the main one, because that does not exist any more. After that change in paradigm happened, I was immersed in a multidisciplinary world. The complex, real world has always been so, but now it is in shades of colour. It is urgent to do a distinct research, because current research is not appropriate and entrepreneurs must loose their fear of academia. Really, we don't bite.

ACKNOWLEDGMENT

The author wishes to express his love and gratitude towards Ixchel Xaman Ek, Akira Itzamaná, and Marulis for their support during the research and composition of this chapter.

REFERENCES

Aaen, I., Bøtcher, P., & Mathiassen, L. (1997). Software factories. In L. Mathiassen (Ed.), *Reflective systems development* (Vol. II, pp. 407-433). Aalborg, Denmark: Aalborg University, Institute for Electronic Systems, Department of Computer Science. Retrieved December 5, 2007, from http://www.cs.auc.dk/~larsm/Dr_Techn/Volume_II/17.pdf

Abrahamson, E. (1991). Managerial fads and fashions: The diffusion and rejection of innovations. *The Academy of Management Review, 16*(3), 586-612.

Aktouf, O. (1998). *La administración entre tradición y renovación*. Cali: Artes Gráficas Univalle-Gaëtan Morin Éditeur.

Alvesson, M. (2004) Knowledge Work and Knowledge-Intensive Firms, New York: Oxford University Press.

Asterion. (2005). Tríptico: Preludio. *Zahir blog*. Retrieved December 5, 2007, from http://www.zonalibre.org/blog/zahir/archives/081847.html

Aubert, N. (1994). Du systeme disciplinaire au systeme managinaire: L'emergence du management psychique. In J.P. Bouilloud & B.P. Lécuyer (Eds.), *L'invention de la gestion. Histoire et pratiques* (pp. 119-136). Paris: L'Harmattan.

Aubert, N., & Gaulejac, V. d. e (1993). *El costo de la excelencia: del caos a la lógica o de la lógica al caos?* Barcelona: Paidós.

Basili, V. R., Caldiera, G., & Rombach, H.D. (1994). The experience factory. In J.J. Marciniak (Ed.), *Encyclopedia of software engineering* (Vol. 1, pp. 469-476). New York: John Wiley & Sons. Retrieved August 5, 2006, from http://wwwagse.informatik.uni-l.de/pubs/repository/basili94c/encyclo.ef.pdf

Ben-Ari, M. (1998). *The software factory*. Paper presented at 10th Annual Workshop of the Psychology of Programming Interest Group, PPIG 1998, Knowledge Media Institute, Open University, UK. Retrieved December 5, 2007, from http://www.ppig.org/papers/10th-benari.pdf

Bertaux, D., & Singly, F.d. (1997). *Les Récits de vie: perspective ethnosociologique* (Collection 128, 122). Paris: Nathan.

Boehm, B. (2006). A view of 20th and 21st century software engineering. In D. Osterweil, D. Rombach, & M.L. Soffa (Eds.), *Proceedings of the 28 th International Conference on Software Engineering ICSE '06* (pp. 12-29). New York: ACM Press.

Boje, D.M. (1999). The storytelling organization game. Retrieved December 5, 2007, from http://business.nmsu.edu/~dboje/between.html

Brooks, F. P. (1995). *The mythical man-month: Essays on software engineering*. Reading, MA: Addison-Wesley.

Brynjolfsson, E. (1994). Information assets, technology, and organization. *Management Science, 40*(12), 1645-1662.

Brynjolfsson, E., & Hitt, L.M. (2000). Beyond computation: Information technology, organizational transformation and business performance. *The Journal of Economic Perspectives, 14*(4), 23-48.

Brynjolfsson, E., & Hitt, L.M. (2003). *Computing productivity: Firm-level evidence* (MIT Sloan Working Paper 4210-01, eBusiness@MIT Working Paper 139). Cambridge, MA: MIT Sloan School of Management. Retrieved December 5, 2007, from http://ebusiness.mit.edu/erik/cp.pdf

Bueno, C. (2000). QS9000: Calidad en la diversidad. *Revista Mexicana de Sociología, 62*(3), 29-49.

Burr, V. (2003). *Social Constructionism*. London: Routledge.

Carmel, E., & Bird, B.J. (1997). Small is beautiful: A study of packaged software development teams. *Journal of High Technology Management Research, 8*(1), 129-148.

Carrillo, J., & Lara, A. (2005). Mexican maquiladoras: New capabilities of coordination and the emergence of new generation of companies. *Innovation: Management, Policy & Practice, 7*(2/3), 256-273.

Clarke, T., & Clegg, S.R. (1998). *Changing paradigms. The transformation of management knowledge for the 21st century*. London: Harper and Collins Business.

Clegg, S.R., & Hardy, C. (1996). Organizations, organization and organizing. In S. Clegg, C. Hardy, & W. Nord (Eds.), *Handbook of organization studies* (pp. 1-28). Thousand Oaks, CA: Sage.

Cusumano, M.A., & Selby, R.W. (1997). How Microsoft builds software . *Communications of ACM, 40*(6), 53-61.

Denning, P.J. (1998). Computer science and software engineering: Filing for divorce? *Communications of the ACM, 41*(8), 128.

DiMaggio, P.J., & Powell, W.W. (1983). The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *American Sociological Review, 48*(2), 147-160.

Dybå, T. (2000). Improvisation in small software organizations. *Software, 17*(5), 82-87.

Dybå, T. (2001). *Enabling software process improvement: An investigation of the importance of organizational issues* (NTNU 2001:101, IDI Report 7/01). Unpublished doctoral dissertation, Norwegian University of Science and Technology, Trondheim, Norway. Retrieved December 5, 2007, from http://www.idi.ntnu.no/grupper/su/publ/phd/dybaa-dring-thesis-2001.pdf

Dybå, T. (2002). Enabling software process improvement: An investigation of the importance of organizational issues. *Empirical Software Engineering, 7*(4), 387-390.

Dybå, T. (2003). Factors of software process improvement success in small and large organizations: An empirical study in the Scandinavian context. *ACM SIGSOFT Software Engineering Notes, 28*(5), 148-157.

Dybå, T. (2005). An empirical investigation of the key factors for success in software process improvement. *IEEE Transactions on Software Engineering, 31*(5), 410-424.

Dybå, T., Kitchenham, B.A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. *Software, 22*(1), 58-65.

Enriquez, E. (1992). *L'Organisation en analyse*. Paris: Presses Universitaires de France.

Ensmenger, N.L. (2001a). *From "black art" to industrial discipline: The software crisis and the management of programmers*. Unpublished doctoral dissertation, University of Pennsylvania.

Ensmenger, N.L. (2001b). The "question of professionalism" in the computing fields. *IEEE Annals of the History of Computing, 23*(4), 56-74.

Ensmenger, N.L. (2003). Letting the "computer boys" take over: Technology and the politics of organizational transformation. *International Review of Social History, 48*(Supplement), 153-180.

Ensmenger, N.L., & Aspray, W. (2000). Software as labour process. In U. Hashagen, R. Keil-Slawik, & A.L. Norberg (Eds.), *Proceedings of the International Conference on History of Computing: Software Issues* (pp. 139-165). New York: Springer-Verlag.

European Commission. (2005). *The new SME definition. User guide and model declaration*. Europe: Enterprise and Industry Publications.

Gergen, K.J. (1996). *Realidades y relaciones. Aproximaciones a la construcción social*. Barcelona: Paidos.

Glass, R.L. (2006). The Standish report: Does it really describe a software crisis? *Communications of the ACM, 49*(8), 15-16.

Goffman, E. (1993). *La presentación de la persona en la vida cotidiana*. Buenos Aires: Amorrortu.

Greenfield, J., & Short, K. (2003). Software factories: Assembling applications with patterns, models, frameworks and tools. In R. Crocker & G.L. Steele (Eds.), *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (pp. 16-27). Anaheim, CA: ACM Press.

Griss, M.L., & Wendtzel, K.D. (1994). Hybrid domain-specific kits for a flexible software factory. In H. Berghel, T. Hlengl, & J. Urban (Ed.), *Proceedings of the 1994 ACM Symposium on Applied Computing 1994* (pp. 47-52). Phoenix: ACM Press.

Hackman, J.R., & Wageman, R. (1995). Total quality management: Empirical, conceptual, and practical issues. *Administrative Science Quarterly, 40*(2), 309-342.

Haigh, T. (2001). The chromium-plated tabulator: Institutionalizing an electronic revolution, 1954–1958. *IEEE Annals of the History of Computing, 23*(4), 75-104.

Heckman, S.J. (1983). Weber's ideal type: A contemporary reassessment. *Polity, 16*(1), 119-137.

Hibberd, F.J. (2005). *Unfolding Social Constructionism*, New York: Springer.

Hodgson, D.E. (2004). Project work: The legacy of bureaucratic control in the post-bureaucratic organization. *Organization, 11*(1), 81-100.

Hualde, A. (2003). ¿Existe un modelo maquilador?: Reflexiones sobre la experiencia mexicana y centroamericana. *Nueva Sociedad, 186*. Retrieved December 5, 2007, from http://www.iztapalapa.uam.mx/amet/debate/modelomaquilador.html

Hualde, A. (2004, August). *Proximity and trans-border networks in the US Mexican border: The making of a software cluster*. Paper presented at the *4th Congress on Proximity Economics: Proximity*. Marseille: Université de la Méditerranée, Groupe de Recherche Dynamiques de proximité. Retrieved December 5, 2007, from http://139.124.177.94/proxim/viewabstract.php?id=82

IEEE Computer Society. (2004). *Guide to the software engineering body of knowledge (SWEBOK)*. Los Alamitos, CA.

Ilavarasan, P.V., & Sharma, A.K. (2003). Is software work routinized? Some empirical observations from Indian software industry. *The Journal of Systems and Software, 66*(1), 1-6.

Institute of Electrical and Electronic Engineers. (1990). *IEEE Std 610.12-1990. IEEE Standard glossary of software engineering terminology*. Los Alamitos, CA.

Keir, M. (1918). Scientific management simplified. *Scientific Monthly, 7*(6), 525-529.

Kuhn, T.S. (1971). La estructura de las revoluciones científicas. Argentina: Fondo de Cultura Económica.

Laporte, C.Y., Papiccio, N.R., & Trudel, S. (1998). *A software factory for the Canadian government Year 2000 Conversion Program*. Paper presented at Software Process Improvement 98, Monte Carlo. Retrieved December 5, 2007, from http://www.lrgl.uqam.ca/publications/pdf/701.pdf

Lethbridge, T.C., Sim, S.E., & Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering, 10*(3), 311-341.

Lewerentz, C., & Rust, H. (2000). Are software engineers true engineers? *Annals of Software Engineering, 10*(1-4), 311-328.

Ludewig, J. (1996). Software engineering: Why it did not work. In A. Brennecke & R. Keil-Slawik (Eds.), *History of software engineering* (pp. 25-27). Dagstuhl Seminar, Paderborn University. Retrieved December 5, 2007, from http://citeseer.ist.psu.edu/rd/0%2C229833%2C1%2C0.25%2CDownload/http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/2159/ftp:zSzzSzftp.dagstuhl.dezSzpubzSzReportszSz96zSz9635.pdf/history-of-software-engineering.pdf

Mahoney, M.S. (1990). The roots of software engineering. *CWI Quarterly, 3*(4), 325-334. Retrieved December 5, 2007, from http://www.princeton.edu/%7Emike/articles/sweroots/sweroots.pdf

Mahoney, M.S. (2004). Finding a history for software engineering. *Annals of the History of Computing, 26*(1), 8-19.

Mauss, M. (1970). *The Gift: Forms and Functions of exchange in Archaic Societies*, London: Routledge.

Mauss, M. (2002[1924]) *Essai sur le don*, Retrieved May12, 2006, from http://socioeconomie.free.fr/MAUSS/essai_sur_le_don.pdf

McIlroy, M.D. (1969). Mass produced software components (and discussion panel). In. P. Naur & B. Randell (Eds.), *Software engineering: A conference sponsored by the NATO Science Committee*. Garmisch, Germany: NATO Scientific Affairs Division. Retrieved August 15, 2006, from http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Meijaard, J., Brand, M.J., & Mosselman, M. (2005). Organizational structure and performance in Dutch small firms (SCALES-paper N200420). Netherlands: EIM Business and Policy Research: Scientific Analysis of Entrepreneurship and SME. Retrieved December 5, 2007, from http://www.eim.net/pdf-ez/N200420.pdf

Miller, D., & Hartwick, J. (2002). Spotting management fads. *Harvard Business Review, 80*(10), 26-27.

Miller, J.P. (1970). Social-psychological implications of Weber's model of bureaucracy: Relations among expertise, control, authority, and legitimacy. *Social Forces, 49*(1), 91-102.

Mintzberg, H. (1975). The manager's job: Folklore and fact. *Harvard Business Review, 53*(4), 49-61.

Mintzberg, H. (1979). *The structure of organizations*. Englewood Cliffs, NJ: Prentice Hall.

Mintzberg, H., & Westley, F. (2001). Decision making: It's not what you think. *MIT Sloan Management Review, 42*(3), 89-93.

Montaño, L. (2005). ¿Qué son los estudios organizacionales?, conference presented at the *Coloquio Internacional: Los Estudios Organizacionales en México: Una Década de Investigación, formación y vinculación*, Universidad Autónoma Metropolitana – Iztapalapa, September 27th, 2005, Mexico City.

Mugler, J. (2004). The configuration approach to the strategic management of small and medium-sized enterprises. In *Proceedings of the Budapest Tech Jubilee Conference*. Budapest: Budapest Tech. Retrieved December 5, 2007, from http://www.bmf.hu/conferences/jubilee/Mugler.pdf

Mutafelija, B., & Stromberg, H. (2003). *ISO 9001:2000 – CMMI v1.1 Mappings*. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute. Retrieved December 5, 2007, from http://www.sei.cmu.edu/cmmi/adoption/pdf/iso-mapping.pdf

Naur, P., & Randell, B. (Eds.). (1969). *Final report from software engineering: A conference sponsored by the NATO Science Committee*. Garmisch, Germany: NATO Scientific Affairs Division. Retrieved December 5, 2007, from http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Organisation for Economic Co-operation and Development. (2002). *OECD small and medium*

enterprise outlook 2002. Paris: OECD Publications Service.

Parnas, D.L. (1998). *Software engineering: An unconsummated marriage*. Paper presented at McMaster University, Hamilton, Ontario, Canada. Retrieved December 5, 2007, from http://www.cs.utexas.edu/users/software/1998/parnas-19981208.pdf

Pinto, J.K., & Mantel, S.J. (1990). The causes of project failure. *IEEE Transactions on Engineering Management, 37*(4), 269-276.

Project Management Institute. (2004). *A guide to the project management body of knowledge (PMBOK guide)*. Newton Square, PA.

Reed, M. (1996). Organizational theorizing: A historically contested terrain. In S. Clegg, C. Hardy, & W. Nord (Eds.), *Handbook of organization studies* (pp. 25-50). Thousand Oaks, CA: Sage.

Rosa, A. d.e la. (2000). La micro, pequeña y mediana empresa en México: Sus saberes, mitos y problemática. *Revista Iztapalapa, 20*(48), 183-220. Retrieved December 5, 2007, from http://148.206.53.230/revistasuam/iztapalapa/include/getdoc.php?rev=iztapalapa&id=656&article=667&mode=pdf

Scott, W.R. (1992). *Organizations: Rational, natural, and open systems*. Englewood Cliffs, NJ: Prentice Hall.

Sharp, H., & Robinson, H. (2004). An ethnographic study of XP practice. *Empirical Software Engineering*, *9*, 353-375.

Sievers, B. (1994). *Work, death, and life itself: Essays on management and organization*. Berlin: Walter de Gruyter.

Simon, L. (2002). *Le management en univers ludique: Jouer et travailler chez UbiSoft, une entreprise du multimédia à Montréal (1998-1999)*. Doctoral thesis, École des Hautes Études Commerciales, Université de Montréal, Montréal, Canada.

Sinclair, B. (2003). Can Mexico develop a software maquiladora industry? *Infoamericas Tendencias Latin American Market Report, 38*. Retrieved December 5, 2007, from http://tendencias.infoamericas.com/article_archive/2003/038/038_industry_analysis.pdf

Smith, J., & McKee, P. (2001). Troubled IT projects: Prevention and turn around (IEE Professional Applications of Computing Series, 3). England: MPG Books.

Stewart, M. (2006, June). The management myth. *The Atlantic Monthly*, *June,* pp. 80-87. Retrieved December 5, 2007, from http://www.edst.educ.ubc.ca/courses/EADM532/Stewart.management.myth.pdf

Thayer, R.H. (Ed.). (1997). *Software engineering project management*. Piscataway, NJ: Wiley-IEEE Computer Society Press. Retrieved December 5, 2007, from http://media.wiley.com/product_data/excerpt/08/08186800/0818680008.pdf

Traxler, F. (2005). Firm size, SME and business interest associations: A European comparison. In F. Traxler (Coord.), *Small and medium sized enterprises and business interest organisations in the European Union*. Europe: European Commission, DG Employment. Retrieved December 5, 2007, from http://www.ueapme.com/docs/projects/Project%20Business%20Associations/study_final.pdf

Tsoukas, H., & Hatch, M.J. (2001). Complex thinking, complex practice: The case for a narrative approach to organizational complexity. *Human Relations, 54*(8), 979-1013.

Tuckman, A. (1994). The yellow brick road: Total quality management and the restructuring of organizational culture. *Organization Studies, 15*(5), 727-751.

Turner, J.R., & Müller, R. (2003). On the nature of the project as a temporary organization. *International Journal of Project Management, 21*, 1-8.

Van de Ven, A.H., & Poole, M.S. (2005). Alternative approaches for studying organizational change. *Organization Studies, 26*(9), 1377-1404.

Winter, M., & Smith, C. (2006). *Rethinking project management (EPSRC Network 2004-2006)* (Final Report). United Kingdom: Engineering and Physical Sciences Research Council. Retrieved December 5, 2007, from http://www.mace.manchester.ac.uk/project/research/management/rethinkpm/pdf/final_report.pdf

Zavala, J. (2004, February). ¿Por qué fracasan los proyectos de software? Un enfoque organizacional. Paper presented at *Congreso Nacional de Software Libre 2004*. February 11th, 2004, Mexico City. , Retrieved December 5, 2007, from http://www.consol.org.mx/2004/material/63/por-que-fallan-los-proy-de-soft.pdf

Zavala, J. (2006). *Dinámica organizacional en el área informática de una organización pública de México.* Unpublished master's dissertation, Universidad Autónoma Metropolitana–Iztapalapa, Mexico City, Mexico.