

Methods as Knowledge Enablers in Software Development Organizations

Mikael Schönström

Informatics, School of Economics and Management, Lund University

Ole Römers väg 6

SE-223 63 Lund, Sweden

Tel: +46 8 56863444

mikael_schonstrom@hermes.ics.lu.se

Sven A. Carlsson

Informatics, Jönköping International Business School

P.O. Box 1026

SE-551 11 Jönköping, Sweden

Tel: +46 36 157504

sven.carlsson@jibs.hj.se

Abstract

Knowledge management (KM) plays an increasingly important role in software development. We find that a missed aspect of software development methods is their possibility to play a role as knowledge enabler. Knowledge enablers are organizational mechanisms for intentionally and consistently developing knowledge in organizations. Drawing on KM-theories we discuss what roles methods can play as knowledge enablers. In two case studies we found that development methods play an important role as knowledge enabler in large software development projects. We found that common development methods stimulate individual knowledge development and facilitate the sharing of individual knowledge and its transformation to organizational knowledge. Methods create a common platform for communication and understanding by defining a communicative framework consisting of common terminology, workflows and best practices. The paper extends theories regarding knowledge enablers and shows that development methods can fruitfully be viewed as a type of knowledge enabler. The paper also discusses how methods can be enhanced to better function as knowledge enablers.

Keywords

Knowledge management, knowledge enabler, software development, software development methods.

1. Introduction

Software-based *innovations* are becoming increasingly important and especially ICT-software is changing innovation processes (Quinn, Baruch & Zien 1997). Software development encompasses a large part of total R&D-spending and, despite falling stock prices, drives much of today's economy. It has proved difficult to produce high quality software on time and on budget (Hilburn, Hirmanpour & Khajenoori 1999), despite the introduction of different software process improvement (SPI) approaches (Osterweil 1987, 1997). The complexity of today's software products is a major cause for making the software development activity and its management a continuously increasing knowledge intensive task. Said Royce (1998): "*The need for software, its breadth of applications, and its complexity continue to grow almost without limits.*"

There has been a strong belief that there is a significant positive correlation between the quality of the software development process and the end product (e.g. Cugola & Ghezzi 1998). Therefore software methodologies, methods, techniques and tools have been at the core of information systems research and practice since the 1960s. Despite the fact that software development projects have to withstand both internal and external uncertainties and changes, methods are still based in that software development is to a large extent a linear process requiring a stable environment (Truex, Baskerville & Travis 2000, Cugola et al. 1998, Fitzgerald 1996). Methods based on the assumption of a stable environment and viewing software development as a phased process have weaknesses like that the success of later phases depends on the success of earlier phases, which requires a "perfect" foresight (Fitzgerald 1996). To have a closed world assumption is not feasible in today's software development projects (Cugola et al. 1998). Thus, it has been argued that there is need to reconsider the role of methods (Fitzgerald 1998, 2000).

The mentioned challenges in software development require new perspectives and approaches on the development activity and its methods. The knowledge intensive nature of software development and its relation to a number of knowledge management (KM) issues and challenges makes software development an interesting phenomenon to study from a KM-perspective. KM is relevant to the area of software development methods because: a) software development is primarily a human and knowledge intensive activity, b) software project faces increased uncertainties in its environment, c) software development is considered to be an innovation intensive R&D practice since software products are usually only developed once with the goal to solve a specific problem (Sheremata 2002).

The objective of this paper is twofold: first, using KM-literature we extend the discussion on software development methods and argue that one missed aspect of these methods is their roles as knowledge enablers. Second, based on a case study and using KM-literature we propose how the development methods can be enhanced to better serve as knowledge enablers in large software development projects.

The remainder of the paper is organized as follows: the next section briefly reviews important methods literature and presents the idea of methods as knowledge enablers. Following, the research method is outlined, and we present our case companies (SAAB TechSystems AB and Ericsson AB) and the case findings. This is followed by discussions, suggestions, and conclusions.

2. Methods and knowledge enablers

“Software engineering is dominated by intellectual activities that are focused on solving problems of immense complexity with numerous unknowns in competing perspectives.” (Royce, 1998). Software development is a complex task that combines many different activities and these activities are performed in an ambiguous context. To develop high quality software on time and on budget has therefore been a challenge (Cugola et al. 1998). In order to take control over the software development process, methods have been proposed as an approach to increase predictability and control of projects as well as to increase product quality. In this paper we will use the term software development, which is a broader term than software engineering. By using development instead of engineering we want to incorporate non-engineering aspects like knowledge management. Another key concept for this paper is the multi-definitional concept method. Here we use method to denote a “recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems.” (Maddison 1983, cited in Avison & Fitzgerald 1995). We have chosen this rather broad definition to enable the usage of knowledge management theories when analyzing methods to go beyond the notion of a method as just a collection of phases and procedures.

The method literature can be divided into two strands: a pro-method and an anti-method. The anti-method strand argues that the use of methods leads to increased development time rather than decreased, there is a too heavy emphasis on document production, and methods are too complex and specify more steps than are needed in most projects. Methods have also been criticized for being inflexible and for having a one-size-fits-all approach, which is problematic since every software development project is unique and different issues (e.g. cultural, political, graphical design, software architecture etc) need to be emphasized in different projects (Avison & Fitzgerald 1999). An additional problem mentioned is that methods do not support the changing nature of software development and still view software development as a linear process.

There are views presented on methods in the literature that are pro-method and also KM-related. Cugola et al. (1998) found that it has been suggested in the literature that software development methods are created to provide expert guidance and wisdom to development processes. Other reasons for using methods throughout an organization are that they are assumed to facilitate change and transfer of staff from project to project without retraining. Additionally, use of methods is believed to facilitate reuse of knowledge and experiences (Avison et al. 1999). “The value of methods and tools, and their conglomerates – methodologies – is that they embody practices and cognitive frames that can be taught, shared and refined over continued trials. Methods can thus be conceived as consisting of directions and rules of action (stocks of knowledge) according to some systematic ordering...” (Hirschheim, Klein & Lyytinen 1996). Methods define the “language” used for different activities and support the development of shared mental models. The use of language and mental models play an important role in knowledge creation and sharing: “Research shows time and again that a shared language is essential to productive knowledge transfer. Without it, individuals will neither understand nor trust each other.” (Davenport & Prusak 1998). Since methods define terms, activities and processes they can create a common ground for how an organization views the development process. Methods can be viewed as part of a knowledge-enabling context. Thus, in the pro-method literature there is a foundation for viewing methods as a mechanism that enables knowledge creation and sharing.

It is proposed in the strategic management literature that a way to create competitive advantage is through superior knowledge (Spender 1996, Nonaka & Teece 2001).

Knowledge creation is thus a key process in this endeavor. Taking away obstacles for knowledge generation is therefore of strategic importance, i.e. identifying and implementing knowledge enabling mechanisms. The creation of knowledge enablers has been emphasized in the KM literature (Ichijo, von Krogh & Nonaka 1998). The argument used for implementing knowledge enablers is that knowledge development in organizations is fragile and should not be left unconsidered. Knowledge enablers are supposed to facilitate consistent and systematic knowledge development. Knowledge enablers are "...organizational mechanisms for intentionally and consistently developing knowledge in organizations." (Ichijo et al. 1998). Ichijo et al. (1998) specify three different roles for knowledge enablers:

- Knowledge enablers should stimulate individual knowledge development.
- Knowledge enablers should protect knowledge development in organizations (i.e. take away obstacles, protect and support knowledge creation).
- Knowledge enablers should facilitate the sharing of individual knowledge and experience among organizational members so that individual knowledge will be transformed into organizational knowledge.

Based on a case study they identify five different knowledge enablers:

- *Knowledge intent*. Establish a sense for knowledge as a competitive resource within the company.
- *Organizational conversations*. Focus on the work to establish a common language within the company, which is commonly shared and understood by the organizational members.
- *Organizational structure*. Implement an organizational design that facilitates knowledge development. (E.g. create an organization that works close to its customers and that have access to various information that can be interpreted differently. This is believed to nurture creativity.)
- *Care relationships*. Manage relationships between organizational members and foster a culture that emphasises patience and tolerance.
- *Knowledge managers*. Stimulate managers that actively collect information, share information and that penetrate different contexts for knowledge creation (e.g. new markets)

The presented theory on knowledge enablers is broad and leaves out many details on how these enablers should be designed. The study was based on a single case, MYCOM, a Japanese company with 2500 employees, manufacturing industrial freezers. The limited knowledge about knowledge enablers and how they function in knowledge intensive environments makes it difficult to plan and to implement these in organizations. It is therefore of importance to further study what type of mechanisms that could function as knowledge enabler in certain contexts. The suggestion that development methods could serve as knowledge enablers in a software development context needs to be explored.

3. Research Methodology

Our purpose is to examine the use of software development methods as knowledge enablers and to increase our understanding of software methods in use. We chose an interpretive case based research strategy. Driven by our interest in finding relevant and interesting research objects we turned to the high-tech industry which is much dependent on software development. We selected two cases that we found relevant and interesting for exploring the

role of software development methods as knowledge enablers. Our case study is therefore partly theory testing since we want to explore if the theory can be used in this context and to extend the theory of knowledge enablers. Further, cases can be chosen according to several rationales, one mentioned by Eisenhardt (1989) is to choose cases that provide examples of polar types, which can extend the use of the theory. We choose one large organization (Ericsson) running projects with over 1000 people in the telecom sector, and one smaller firm (SAAB) developing military applications and where projects often had the size of 100 people.

We did interviews and used also internal documents. In the Ericsson case we also drew on personal experiences from methods and tools development and the use of these methods and tools in real projects. The documents studied were method descriptions and project specifications for projects using the methods. We interviewed 15 persons. The interviewees had roles such as programmers, project managers, systems architects, project planners, configuration managers, and in one of the organizations we also interviewed an IS/IT executive. Ten interviews were done at SAAB TechSystems AB and five at Ericsson AB. The limited number of interviews at Ericsson can be explained by the fact that one of the authors is also an employee at Ericsson working with software development. The interviews were done during April and May, 2002, and the interviews were face-to-face semi-structured interviews and each interview ranged between 45–90 minutes. They were tape-recorded and transcribed. The questions in the interviews focused on the use of methods in software projects, what problems the projects faced, how the problems were addressed and solved, how project-related knowledge and experiences were captured, and what roles the methods played in relation to knowledge creation and sharing.

4. Case Findings

4.1 The Cases

SAAB TechSystems AB –a company in the SAAB Group–develops, manufactures, and maintains military command and control systems and combat management systems as well as commercial systems. The company has about 750 employees located in Stockholm, Uppsala, Härnösand and Adelaide (the latter an Australian subsidiary). In Sweden SAAB TechSystems consists of 5 divisions. The division in this study was Naval Systems. They have mainly larger customers (national defense forces) and project size is often around 100 people. The project-staff work together from start to finish, which is often a period of about 2–3 years. Changes in project teams, i.e. switching project members during the course of a project were uncommon. Quality was an important factor since they develop systems used in extreme situations where an error can put lives at risk. The developed systems are used for a long time, sometimes up to 15–20 years. In most cases the systems have to meet international quality standards, like the US. Department of Defense's (DoD) military standards (MIL).

Ericsson AB is the Swedish part of the Ericsson Group. The Ericsson Group has about 76,000 employees in 140 countries and Ericsson AB has around 20,000 employees in various locations in Sweden. Ericsson develops, sells, and maintains telecom systems to operators all over the world. The projects in this study were working with software development for the 3rd generation (3G) mobile networks. The interviews spanned different units and projects involved in software development. The development of the different releases of the mobile networks is organized in a complex project hierarchy. A total project for one release of a network can include between 2,000 and 5,000 people—in some cases even more people. The

products are very complex due to the large number of components and sub-systems that must work together. Project management and organization was complicated by that the projects were geographically and organizationally distributed. The development activities were spread to five or six different countries. The organizational turbulence the company went through during our study with large layoffs, re-organizations, and a turbulent telecom market added to the complexity of the company's environment.

4.2 Methods

The development methods in both organizations were documented in either Word or HTML. The method documents included both text descriptions and workflow diagrams and other types of conceptual models showing software development activities from a process perspective. The studied methods covered general technical management aspects of the projects rather than technical details for specific products. The methods had more of an engineering focus than an organizational management focus.

SAAB had a specific set of methods that were used in their projects and the set of methods was well documented and well established in the organization. The methods did not have any thorough or detailed specifications on *how* things should be done, instead they contained frameworks that explained *what* should be done and *why*. It was stated that the methods were part of the organization's quality system. The methods were based on several international standards from ISO, IEEE, MIL and TickIT.

Ericsson has many local design centers and it was common that these had developed their own methods. This complicated the situation, especially in the large 3G-projects where many units that hadn't worked together before, should start working together with a mixed set of methods. The various methods used slightly different terminology or sometimes the same terminology but with different meanings. The current technology shift in mobile communication, from 2G to 3G, implied many changes at Ericsson and created often a whole new situation for the project members. New methods and tools had to be developed and implemented as the projects went on. Existing methods were simply not suitable in the new environment and they had to be updated or replaced with new methods. 3G-projects had often to start without a stable methods and tools environment. The projects had to create new methods defining necessary concepts that better explained the new context.

4.3 The project work

The type of software developed by the two companies varied. SAAB developed advanced combat management systems that often were custom-made. Ericsson developed also large systems but with the aim that these could be sold to several operators having different configurations. Even if the products developed and the size of the projects differed between the cases, the work in the project, sub-projects etc did not differ to any significant extent.

Software development in both case organizations was organized in projects. Larger projects were called programs and consisted of projects, which in turn consisted of sub-projects. In some cases the larger sub-projects could be viewed as communities of knowing (Boland & Tenkasi 1995) as they had developed local work practices and had specialized knowledge.

Organizational and technological changes were common in both organizations, which affected the design of methods. Some methods needed to be re-designed to fit the new situations and for enabling an efficient communication. A project's organizational complexity increased with product complexity, and it was very difficult to get an overview

over the project organization in the largest projects. It also seemed that the technical complexity to a certain point stalled, regardless of how large a project was. The complexity in the largest projects was related more to planning and coordination of all the tasks that had to be completed rather than being related to technical issues and problems.

Common in both organizations was the problem of delayed projects. There were many reasons mentioned to this problem, everything from “lazy” project members to the complexity of the products to be produced. The complexity made it very difficult to up-front estimate how long time it would take to develop a system. It was almost impossible for anyone to foresee all the dependencies of the components to be developed and how they would work together in the final system. The projects in both companies were run over very long periods of time, usually at least 2 years. Organizational changes, like re-organizations in the line organization, mergers between companies etc made the organizational environment for the projects unstable in both cases.

4.4 The use of methods in projects

In both organizations the exchange of experiences and knowledge between the M&T organization (the unit responsible for methods development) and the projects were rare. Communication was more dependent on personal relationships and agendas rather than organizational procedures. Therefore the M&T organization rarely received any feedback from the projects, thus having problems in understanding how the methods were used in the projects, and how the methods could be improved. Interviewees in the projects (developers and project managers) rarely knew how or to whom suggestions for changes in the methods should be communicated. The interviewees believed that better procedures for capturing project experiences in the end of the projects could support future projects in the start up phase.

It was common to customize the corporate methods for each project. The level of customization varied. Some projects made significant customizations, e.g. specifying how a method should be used in cooperation with a tool, others just set up a project web site with methods pages pointing to the corporate method and made only brief comments with regard to the project. The knowledge and the awareness of the corporate methods varied depending on the level of experience among the project members. Individuals that had a long working experience with the company knew less about the methods or had difficulties in expressing what the methods actually specified and prescribed. These individuals had worked in the organization for such a long time that they had more or less “internalized” the methods. The methods played an important role for the new employees since they needed a “framework” for making sense of their new environment. The method provided them with a point of reference and some basic terminology, which made it easier to spot lack of knowledge and facilitated communication with more experienced colleagues.

All methods had a terminology section, either in the form of a glossary or a special section where key concepts were explained and discussed. These “features” in the methods that focused on the use of a common language showed to have a very useful impact. As one interviewee said “Even if there is a lot of talking and intense communication within the project, everyone has his/her own interpretation which often means that an activity may not fit in the whole. In small projects this is not a problem, but if you are about a hundred then the problems with different interpretations become severe.” The glossaries were of extra value in the large Ericsson projects that had several sub-projects geographically dispersed. The terminologies ensured the least common denominator for communication, thus facilitated communication and sharing over national and cultural borders.

5. Discussion and Conclusions

The studied software development projects faced many changes during the course of the projects due to both internal and external changes. In our study, a natural cause for this was that many projects run over a long period of time, 2-3 years, and organizational changes and technological changes are common in high-tech industries. Operations in dynamic environments require learning and flexibility. Methods that specify the minimum level of commonality are essential in this sense since they make collaboration possible, but leave enough room for software teams to solve innovation problems (Cugola et al. 1998). Flexible methods are a prerequisite in the fast moving environment of software development and customizations of existing methods were the norm in the case companies. Each development project is a unique undertaking and therefore methods need to be customized to support the specific task (as argued by e.g. Fitzgerald 1998). Sometimes major changes require that entirely new methods need to be developed that incorporates new terminologies and workflows based on new assumptions about a project's environment. This was evident at Ericsson, where the major technology shift demanded new ways of working and thus new methods. We argue that product and project complexity and environmental turmoil emphasize the importance for knowledge management and the use of common methods as knowledge enablers in the software development industry.

An important role the methods played in our case companies was to facilitate access to knowledge by providing a common platform for collaboration between project teams, as the methods defined important terminology and workflows. This feature was especially important in the large distributed organizations. The methods provide a common base for discussion and understanding. If projects or other development units create own methods that are not based on a common corporate method there is a great risk that workflows and terminologies will differ to such an extent that it is difficult to have effective inter-project communication. The case study suggests that software development methods will serve as a knowledge enablers, both in large and in small projects, but the size of the project has an impact. In the small single-project environment the method enables communication and understanding *within* the project. If different methods, based on different assumptions and rationales, are used in a multi-project environment they will *disable* communication and understanding between projects. For large projects that consist of many sub-projects it is therefore important that a *common* method is used to enable communication and understanding between projects working together. Ichijo et al. (1998) emphasize the importance for companies to focus on the development of a *common language*, which will facilitate understanding and knowledge sharing. An important finding in our case study was that if no common method was implemented in the development organization, capturing of knowledge and experiences from projects were obstructed. The structures or frameworks for knowledge and experience capturing were missing and left out to personal networks, which are hard to access for e.g. new employees. Our study showed that methods were useful to new employees since they by the help of the methods could more rapidly understand what knowledge they lacked. Methods facilitated also communication with more experienced people since the method provided the new employees with a terminology. Knowledge management is not only to capture new knowledge and to re-use it in new contexts but also to take away obstacles for communication to enable an efficient communication (Sheremata 2002). Our study showed that the methods have an important role to play in this regard.

In a software development context that is characterized by a complex project hierarchy of projects and sub-projects, we argue based on our study, that a common method play all the knowledge enabling roles mentioned by Ichijo et al. (1998).

- Methods *stimulate individual knowledge development*. New employees can use methods to identify what they need to know more about and they facilitate conversations with more experienced developers.
- Methods *protect and support knowledge development*. They take away obstacles for communication and provide the organization with a communicative framework and contribute to knowledge development that is non-random and unsystematic.
- Methods *facilitate the sharing of individual knowledge and its transformation to organizational knowledge* since they contain common terminology and workflows, thus creating a common ground for communication

Regarding the five *types* of knowledge enablers proposed by Ichijo et al. (1998), we found that the development methods in our cases only map the type of *organizational conversations*. A method is a type of knowledge enabler that establishes a common language and understanding within software development projects. A common method work as a platform for intra-organizational communication. Boland et al. (1995) argue that knowledge intensive firms are composed of multiple communities of specialized knowledge workers (communities of knowing). These communities may become incommensurable over time as values, perspectives and language develop to be unique for that community. We suggest that methods play an important role to overcome the incommensurability between communities of knowing.

To strengthen and extend the role of methods as knowledge enabler in a software development organization we propose that method developers also should aim at implementing the knowledge enabler *knowledge intent* in the method. The method can e.g. contain a section that gives an explanation why knowledge is important to the project and the organization. This knowledge intent can then be complemented with procedures for how knowledge is captured in projects and fed-back to the organization. To a software development organization we suggest that methods is a specialized type of knowledge enabler consisting of the two generic knowledge enablers *knowledge intent* and *organizational conversations*.

We found that the KM literature plays an important role in enriching the software development domain and that it can increase and extend our understanding for software development as knowledge work. In our paper we concentrated on the role of methods as knowledge enablers. We found support in the literature for viewing methods as knowledge enablers in software development work. We also found that the theories regarding knowledge enablers were rather vague and broad. Via our case study we were able to extend the theory of Ichijo et al. (1998). We found that the theory of knowledge enablers is useful in a software development context and that the software development method functions as knowledge enabler in this context. Methods are important knowledge enablers and should therefore receive greater attention as knowledge enablers by both managers and systems developers.

A limitation of this study is that we did not explore exactly how a method should be designed to play a knowledge-enabling role. Even if we argue that terminologies and workflows help projects to understand and to communicate, we can't expect that *any* method containing terminology and workflows will play an enabling role. There might be specific method "features" in combination with certain project characteristics that have a role in deciding if a method plays a knowledge enabling role or not. For future research we suggest further research regarding these knowledge-enabling features in methods, and on finding other types of mechanisms that can play the role of knowledge enabler in the software development organization.

References

- Avison, DE & Fitzgerald, G (1995), 2nd edn, *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill, London.
- Avison, DE & Fitzgerald, G (1999), 'Information systems development', in *Rethinking Management Information Systems*, Currie, WL & Galliers, B (eds), pp. 250-279, Oxford University Press, Oxford.
- Boland, RJ & Tenkasi, RV (1995), 'Perspective making and perspective taking in communities of knowing', *Organization Science*, Vol. 6, No. 4, pp. 350-372.
- Cugola, G & Ghezzi, C (1998), 'Software processes: a retrospective and a path to the future', *Software Process Improvement and Practice*, Vol. 4, pp. 101-123.
- Davenport, T & Prusak, L (1998), *Working Knowledge*, Harvard Business School Press, Boston.
- Eisenhardt, KM (1989), 'Building theories from case study research', *Academy of Management Review*, Vol. 14, No. 4, pp. 532-550.
- Fitzgerald, B (1996), 'Formalized systems development methodologies: a critical perspective', *Information Systems Journal*, Vol. 6, pp. 3-23.
- Fitzgerald, B (1998), 'An empirical investigation into the adoption of systems development methodologies', *Information & Management*, Vol. 34, pp. 317-328.
- Fitzgerald, B (2000), 'Systems development methodologies: the problem of tenses', *Information Technology & People*, Vol. 13, No.3, pp. 13-22.
- Hilburn, TB, Hirmanpour, I, Khajenoori, S, Turner, R & Qasem, A (1999), *A software engineering body of knowledge*, Version 1.0, CMU/SEI-99 TR-04.
- Hirschheim, R, Klein, KH & Lyytinen, K (1996), 'Exploring the intellectual structures of information systems development: a social action theoretic analysis', *Accounting, Management & Information Technology*, Vol. 6, No.1-2, pp. 1-64.
- Ichijo, K, von Krogh, G & Nonaka, I (1998), 'Knowledge enablers', in *Knowing in Firms – Understanding, Managing and Measuring Knowledge*, von Krogh, G, Roos, J & Kleine, D (eds), pp. 173-203, Sage, London.
- Nonaka, I & Teece, DJ (eds) (2001), *Managing Industrial Knowledge: Creation, Transfer and Utilization*, Sage, London.
- Osterweil, L (1987), 'Software processes are software too', in *Proceedings of the Ninth International Conference on Software Engineering*, March 30 - April 2, Monterey, CA.
- Osterweil, L (1997), 'Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9', in *Proceedings of the 19th International Conference on Software Engineering*, May 17 - 23, Boston, MA.
- Quinn, JB, Baruch, JJ & Zien, KA (1997), *Innovation Explosion*, Free Press, New York.
- Rainer, A & Hall, T (2001), 'An analysis of some 'core' studies of software improvements', *Software Process Improvement and Practice*, Vol. 6, pp. 169-187.
- Royce, W (1998), *Software Project Management – A Unified Framework*, Addison-Wesley, Boston.

- Sheremata, WA (2002), 'Finding and solving problems in software new product development', *The Journal of Product Innovation Management*, Vol. 19, pp. 144-158.
- Spender, JC (1996), 'Making knowledge the basis of a dynamic theory of the firm', *Strategic Management Journal*, Vol. 17, Winter Special Issues, pp. 45-62.
- Truex, D, Baskerville, R & Travis, J (2000), 'Amethodical systems development: the deferred meaning of systems development methods', *Accounting, Management & Information Technology*, Vol. 10, pp. 53-79.