SYSTEMS DEVELOPMENT METHODOLOGIES: A KNOWLEDGE PERSPECTIVE

WARREN KERLEY AND TONY HOLDEN
Cambridge University, UK

Abstract. Structured methodologies have for some time been the dominant guiding means by which information systems have been designed and implemented. This paper argues that the process-based view that has underlied information system design (ISD) for so many years could usefully give way to a knowledge-based perspective since ISD is fundamentally a knowledge structuring activity. This paper includes a review of the evolution of systems development methodologies since the late 1960s and contrasts the intended benefits and problems in practice of using structured, process-oriented methodologies such as SSADM, with those of the agile methodologies, such as XP, which have emerged in the last few years. A framework is presented of how ideas from the fields of knowledge management and organizational learning can be applied to the analysis of ISD methodologies. We consider some higher-level considerations of possible future trends in ISD and our analysis suggests that the present preponderance of structured methodologies will give way to greater use of agile and open source approaches. We finish with a description of the work we have done to apply these ideas within current practice and make suggestions for further work in this area.

1. Introduction

Developers of computer-based information systems (IS) are expected to deliver working systems that meet their customers' requirements and are constructed to acceptable levels of quality, maintainability, dependability, efficiency and usability. They are also expected to do this in a way that is cost effective, timely, manages the levels of risk to which the development project is exposed and is flexible enough to incorporate changes throughout the project lifecycle.

In practice this is difficult. It is an established part of ISD wisdom that many systems are delivered late, over budget and behind schedule. Others are cancelled without ever being completed. Even if a system is delivered within time and budget constraints, it may still be perceived as a failure if

J.S. Gero (ed.), Design Computing and Cognition '06, 163–181.

does not meet the expectations of key stakeholders. Moreover, the productivity of software developers has failed to keep pace with the spectacular fall in cost and increase in performance of computer hardware.

2. Methodologies

Researchers and practitioners have not been lax in meeting this challenge. Methodologies have proliferated over the last 30 years in order to organize the activities involved in information systems development. In the 1960s and 70s information systems were developed largely on an ad hoc basis, without any formal methodological support. Driven by the need to manage increasingly large and complex projects, efforts were made to formalize matters and, in the late 1960s, the systems development lifecycle (SDLC) view of software project development, commonly known as the waterfall model, emerged. Many methodologies follow the SDLC approach today due to the support of national governments and organizations such as the US Project Management Institute. Nevertheless by the late 70s and early 80s new approaches based on prototyping and evolutionary development had appeared in response to practitioner experience that showed SDLC methodologies could take too long and be too rigid under some circumstances.

Other work by academics in the early 80's aimed to address the social and sense-making issues involved in ISD and resulted in, for example, ETHICS and the Soft Systems Methodology (SSM). Subsequent work combined the best elements of the existing methodologies, such as merging of prototyping and evolutionary development ideas with the SDLC to produce iterative and spiral methodologies.

The proliferation of methodologies reflected a number of different – and rarely stated – philosophies, assumptions or beliefs about the nature of ISD. These can nevertheless be broadly classified in a number of different ways. For example Avison and Fitzgerald (2003) suggest seven major themes or approaches – structured, data-oriented, prototyping, object-oriented, participative, strategic and systems – which are not necessarily mutually exclusive.

Although supplemented in recent years by visual or object-oriented methods for interface design and specialized systems, the majority of formalized methodologies in the 1990s were still underpinned by structured SDLC and/or prototyping process models. They are therefore still primarily process-oriented and plan-driven with their methods and process models based on concepts that originate in mainstream engineering disciplines. Projects are heavily documented and require strict adherence to the prescribed processes.

Despite all this being highly time and effort consuming, documentation does aid in realizing a degree of quality, control and, importantly for most software companies, reassurance for potential and repeat customers.

Organizations using these approaches consequently look for ways to continuously improve these processes through adherence to quality standards such as ISO certification and process improvement initiatives such as SEI's Capability Maturity Model (CMM).

From a management viewpoint, subdivision of the development process has definite advantages. It reduces the skill-levels required by developers and standardization makes these skills more interchangeable. These are seen as critically important benefits given the shortage and high turnover of software developers (Riemenschneider 2002). Standardization also facilitates project management and control, thereby reducing risk and uncertainty. The large amount of documented information within the process increases management flexibility, as personnel can be moved quickly within or between projects, and provides insurance against the loss of critical knowledge if key personnel leave. These benefits have prompted national governments to encourage the use of structured methodologies and the subsequent support of formal certification and standards such as ISO 9000, ISO 12207, ISO 15504, SSADM within the UK and the CMM by the US Department of Defense.

In practice, however, the use of formalized, process-oriented methodologies has not been without problems. In a case study of a project using SSADM, Wastell (1996) found that the methodology was followed in a blind, mechanical way. The effort spent on the aesthetics of the diagramming techniques, and in providing the detail required by the documentation standards, bogged the project down and provided too much information. The "big picture" was obscured as far as the user representatives were concerned. Process-oriented methodologies work best when the requirements of the software project are completely locked in and frozen before the design and software development commences. However, in an increasingly volatile business environment, firms are asking for lighter weight, faster and more agile software development processes that can accommodate the inevitable ongoing changes to requirements. At the same time there has been a backlash from the programming profession against the mechanistic and dehumanizing aspects of process-oriented methodologies and a desire for a return to programming as a craft rather than an industrial process. These factors have led to the development by practitioners of agile methodologies in the late 1990s, based on prototyping and rapid application development approaches (Abrahamsson et al. 2002).

eXtreme Programming (XP) has been widely acknowledged as the starting point for the various agile software development approaches and is probably the best known. Other agile methodologies include Scrum, Dynamic Systems Development Method, Crystal Methods, Feature-Driven Development and Adaptive Software Development.

No clear agreement has been achieved on how to clearly distinguish agile software development from more traditional, process-oriented approaches. A central tenet of the agile philosophy is that it is impossible to get software

right first time and it is therefore preferable to be responsive to changing customer requirements and to provide them quickly with what they want. Common, although idealized, features of the agile approach are that software development is:

- Incremental with small functional differences and shorter times between releases.
- Cooperative with customer and developers working constantly together with close communication,
- Straightforward in that the method itself is easy to learn and to modify,
- Adaptive and so changes to requirements may easily be included,
- Well, but minimally, documented.

It is this last aspect that most characterizes agile methodologies. It could also be argued that agile methodologies are "more honest" as their approach mirrors more closely the reality of software development in practice.

Concerns have been raised about the use of agile methodologies (Abrahamsson 2002; Boehm and Turner 2003). They do not provide the familiar management control mechanisms and high quality assurance inherent in process-oriented methodologies, and are therefore considered risky. There are also serious doubts about how scalable agile methodologies are to larger projects. Nevertheless experience has shown that both process-oriented and agile methodologies have a role in contemporary software development.

That said how should a project manager choose which methodologies or methods to adopt? No single methodology can work for all types of project. Guidance is needed for practitioners about the methodologies or specific methods that are applicable in a particular circumstance and how to select the best one. The early view that there might be a single best method for all IS development has given way in recent years to investigation of domain-specific methods (Barry and Lang 2002). Although a single methodology may be appropriate in some circumstances, no methodology covers all aspects of systems development and each methodology tends to be stronger on some aspects than others. Many problem situations will require developers to use methods taken from different methodologies. However it requires considerable skill on the part of developers and managers to be able to pick and choose between methods and apply them effectively to the task in hand.

Besides the problem of selecting from the vast numbers available, methods from different methodologies may be incompatible with each other because they are based on different philosophical assumptions or emphasize different modeling stances and therefore representations of the system being considered. For example, Stephens and Rosenberg (2003) describe in detail the dangers of using some, but not all, of the twelve practices of eXtreme Programming.One solution is a methodological framework such as Mulitview that provides a coherent method to choose appropriate methods,

tool and techniques contingent on the problem, the methodology and the information systems development team itself (Avison and Wood-Harper 1991). More recently Boehm and Turner (2003) have outlined a contingency method to identify the parts of a project that are amenable to agile methodologies and those suitable for a process-oriented approach.

3. Use and Benefits of Methodologies

After all the theorists and academics have had their say, are methodologies actually used in practice? Many companies do not use commercially available methodologies at all or extensively modify them to better fit their own particular organizational needs. The scale of this phenomenon means that in-house methods predominate over the formalized methods prescribed in the literature (Barry and Lang 2002). Ironically, little is known about the nature of "homegrown methodologies" or how they are developed.

Where formal methodologies are used, they are rarely followed closely. For example, the pressure of deadlines often leads to practices being modified or abandoned for the sake of expediency (Wastell 1996; Curtis et al. 1988). Estimates vary but between 50 and 75 per cent of US organizations would be classified at CMM Level-1. Namely, immature software organizations in which development is inconsistent and methodologies are not used (Riemenschneider 2002). In one UK survey 60 percent of respondents did not use a development methodology and only 14 per cent claimed to use a formalized commercial methodology, Table 1. "The predominant reason for non-use cited by respondents was that currently available methodologies did not suit the profile of the development prevailing in the organizations studied." (Fitzgerald 2000).

TABLE 1. Methodology Usage.

	%
Organizations not using any methodology	60
Organizations using a formalized commercial methodology	14
Organizations using internal methodology based on a commercial one	12
Organizations using internal methodology not based on a commercial one	14

Secondly, do methodologies work? Adopting a new methodology is a costly and radical step. It involves significant organizational changes; substantial investments in technology, training and staff time; and the need to overcome resistance to change by developers and other stakeholders (Abrahamsson 2002; Riemenschneider 2002; Wastell 1996). Barry and Lang (2002) found that information systems developers appear to be reluctant to abandon older techniques, even when their usefulness may be questionable, and are slow to adopt the new techniques. In this context it seems unlikely

that IS departments will want to invest in multiple methodologies unless they are likely to be effective.

Unfortunately most work to date has been focused on developing new methodologies rather than evaluating their efficacy in practice. Where empirical research has been done the results can be equivocal or contradict other studies. Glass (1999) reviewed the research on a number of new technologies – including structured methodologies – that were expected to bring significant improvements in software development productivity. Fourth generation languages (4GLs) and object oriented (OO) approaches seem to have provided the biggest productivity improvements, although Glass has reservations about endorsing them.

As for structured methodologies, despite their longevity, research into their benefits was surprisingly scarce and could point to, at best, modest benefits from their use. The same lack of hard empirical evidence for their benefits applies to agile methodologies (Abrahamsson et al. 2002), although this may clearly be due, at least in part, to their relative newness. The case studies that have been published on the use of agile methodologies have often shown spectacular improvements in productivity and quality, but these results have been greeted with skepticism by other practitioners and academics who cite the absence, to date, of sufficient data (Boehm and Turner 2003). Third, and finally, are methodologies relevant for the future? Formalized, process-oriented methodologies were seen as the solution to the "software crisis" but have not delivered the benefits expected (Glass 1999). Structured and agile methodologies are based on concepts that came to prominence in the decade between 1967 and 1977 Since then the pace of business change has increased significantly, short-term needs dominate and the economic justification for formalized systems development with its long development lifecycle is dwindling (Boehm and Turner 2003). Systems development is increasingly outsourced or based on the customization of packaged software.

And it has been long recognized that methodology is less important than the skill and determination of developers. The majority of methodologies in use follow a rational, scientific paradigm where information systems development is conceptualized as an orderly process that is amenable to the same sorts of methods as mainstream engineering. In fact practice shows that systems development is anything but rational and orderly and often too little attention is paid to 'softer' social aspects and to human factors such as creativity, intuition and learning over time.

In summary, IS managers are faced with a wide choice of possible methodologies and, although contingency approaches to select methodologies have been proposed, this choice has been further complicated by the agile vs. process-oriented debate. In some respects however the academic and practitioner literature gives greater importance to the formalized, published methodologies than their use in practice warrants, as they are often either not used or are heavily customized in practice. There

are also doubts about their current efficacy and their usefulness in the future given the rapidly changing business environment.

It must be acknowledged however that the use of structure and formality over the last couple of decades has contributed in some way to the delivery of large, complex and functioning software systems. Rather than invent yet another methodology, the following sections take a different, knowledge-oriented perspective on the task of software development. From this view how can the ideas from the knowledge management and organizational learning literatures contribute to a better understanding of the practice of ISD and therefore which methods, tools and techniques should be most effective in delivering IS projects in any given organizational situation?

4. A Knowledge Perspective

Brooks (1987) identified four essential difficulties (or essences) inherent in developing software that make information systems development different from mainstream engineering disciplines:

1. *Complexity:* Software systems have a very large number of different states that increase more than linearly with an increase in system size. This complexity creates problems of communication and understanding, testing and verification, use, reuse, modification and maintenance.
2. *Conformity:* Software is generally expected to conform to the needs of the organization and not vice versa.
3. *Changeability:* Software is highly malleable and therefore successful software will be changed, either at user request or to be used elsewhere.
4. *Invisibility:* Implemented software is invisible. It is also difficult to visualize as multiple modeling techniques are required to fully represent its function and structure. There is also no representational single point of reference. There is no software equivalent of the floor plan of a building or the circuit diagrams and mechanical drawings in engineering.

Brooks argues that no breakthroughs in ISD productivity and quality are likely through the use of orthodox tools and techniques, because these fail to address these four essences. Instead organizations should either avoid the problems by using commercial off-the-shelf software or concentrate on producing great designs from which to develop their systems. Great designs however require a full understanding of system requirements and great designers. According to Walz et al. (1993), it is reckoned that more than half the cost of the development of complex computer-based information systems is attributable to decisions during requirements specification and design. Curtis et al. (1988) found that on large projects the three biggest problems affecting the design phase were: the thin spread of application domain

knowledge amongst developers; fluctuating and conflicting requirements and, communication and coordination breakdowns. Consequently much of the activity on projects is concerned with human skills such as learning, communication and negotiation. Individuals have to acquire and integrate knowledge from multiple domains and from different parts of the project. This learning modifies the participants understanding of the solution and can lead to changes in requirements, design and implementation throughout the project lifecycle.

We suggest that a knowledge perspective, drawing on experience and insights into knowledge management and organizational learning, provides substantial help for the both the effective management of information systems development and also the role that methodologies play in this. Cognitive theories provide an understanding of how individuals handle mental tasks and therefore how the application of ISD methods, practices and tools could be aligned to take proper account of individual factors (Robillard 1999). These factors include: the importance of learning and previous experience and how this contributes to the creation of mental schemas as models for comprehension; the problems inherent in solving ill-defined problems such as those that typically arise during design; the limitations of short-term memory and hence the importance of breaking tasks down – the famous '7±2 chunks' maxim – and how the information resources that surround the ISD team can be organized, rendered accessible and kept track of in order to facilitate good design and implementation decisions.

At the same time, individuals have to work within a larger social and organizational environment (Curtis et al. 1988). Nonaka and Takeuchi's SECI model (1995) provides an explanation of the social process by which knowledge is created and shared within an organization. Skillfully applied, Nonaka's model can inform organizational design and engender the individual learning and effectiveness described above. Central to the model is the notion that individually held tacit knowledge becomes universally held explicit knowledge through a managed process of 'socialization-externalization-combination-internalization'. This requires the careful instigation of key knowledge-transmitting relationships, the explicit expression of knowledge for group consumption and then its wider dissemination within the organization for the enrichment of other individuals' capabilities. These SECI cycles follow each other and provide an adaptable, and at least partially controllable, means to develop, transmit and apply knowledge about aspects of project that is itself changing. The SECI model emphasizes the value of integrating knowledge from various domains (both internal and external to the project) and so mitigates against the establishment of organizational 'silos' whose insularity hinders agility and directs individual energies towards the maintenance of methodological mechanisms rather than project success.

Successful systems development critically depends on users and designers learning from each other. Hence, managed, timely participation in the learning cycles, such as those described by the SECI model, is important. For example, when new members are added to teams existing members may be reluctant to use or accept the new knowledge the newcomers bring with them (Walz 1993). Conversely, how can early project participants influence later stages of a project if they are no longer actively involved? If knowledge is 'captured' once – for example user requirements gathered early in a project – and pushed into the background, it will often have less weight than the knowledge readily and currently available to the team.

With orthodox, methodologically-driven project management, a risk is that achievement of bureaucratic targets and performance measures takes precedence over the delivery of working systems and the satisfaction of end-users. In other words, the emphasis is on the process with its explicit measurement parameters and not the artifact. Knowledge management, properly applied, focuses on the more tacit and inherent qualities of the individuals, who deliver the artifact. This is not to say that performance measures or targets must be eschewed in favor of some set of loosely-defined or lofty and impractical aims. KM brings the focus back to the efficient delivery of a useful artifact and is more amenable to engaging with the communication dynamics and shorter timescales of modern ISD projects.

The literature on the learning organization (the entity) and organizational learning (the process) is a rich source of insight into the way knowledge is created and communicated. For example, Crossan's et al. (1999) model is an explanation of how the creation of new knowledge (feed forward) and the transfer of existing knowledge (feedback) occur through four processes – intuiting, interpreting, integrating and institutionalizing – at individual, group and organizational levels. Montoni et al. (2004) present an approach for acquiring and preserving ISD knowledge and making it available between organizations.

Knowledge work is not programmable and this creates challenges for managers predisposed to command-and-control management. Drucker (1999) asserts that the only way to increase the productivity of knowledge workers is to radically change the way that they are managed:

1. Non-value added activities should be completely eliminated;
2. Knowledge workers should be responsible for their own productivity;
3. Continuous innovation and learning are required, coupled with knowledge workers teaching others what they know and what they can do;
4. Quality of work is at least as important as quantity; and
5. Knowledge workers own the means of production so should be treated as an organizational asset not a cost. Managers, however, will experience tension between allowing employees autonomy and wanting to maintain control of project outcomes.

These issues have long been recognized in ISD and practice has been to grant IT professionals greater autonomy than many other business functions. Management of ISD therefore involves a relatively high degree of professional trust. Nevertheless software development almost always needs to be disciplined if it is to meet its stated objectives – i.e. allowing freedom and responsibility but within a guiding framework. Practitioners and managers alike see the benefits of methodologies in providing structure to the development process (Barry and Lang 2002) and something to manage against. Even if formalized methodologies are not used they still influence practice (Avison and Fitzgerald 2003). Methodologies also provide an organizational framework, a common language, shared paradigms and approaches to problem solving and task completion that help project participants to communicate, cooperate and learn (Wastell 1996).

In contrast, approaching ISD as a knowledge-based activity suggests that any guiding principles employed should attend to the cognitive needs of individuals, the behavioral and social realities of team based working, the desirability of individual and organizational learning and the business needs of flexibility and speed but at an appropriate level of risk.

5. Analysis of Methodologies

In order to analyze a methodology it can be considered in terms of its constituent model, techniques, tools, scope, outputs, users and practice. Also, any methodology will have an underlying intellectual framework or philosophy and will be directed towards a particular application area. It will therefore have particular management objectives and success measures.

How can the two approaches of methodologically- and knowledge and organizational learning- based views on ISD be unified in a way that meets the needs of all parties in the current environment? In the next paragraphs we propose a framework, Figure 1, that relates the two perspectives. We then highlight and discuss cross-linkages between the two.

Just as methodologies are characterized by different underlying philosophies, the knowledge management literature is broadly divided ontologically and epistemologically into objectivist and subjectivist standpoints (Ortenblad 2002). The objectivist standpoint is common in the information systems literature: knowledge is an object that can be separated from both knower and context and much valuable knowledge can be codified, stored and transmitted using information technology. This has led to the development of knowledge management tools to support ISD such as design rationale and experience factories. Kettunen (2003) follows an objectivist philosophy when he proposes methods to manage software development knowledge that involve auditing each person's knowledge and then determining with whom they should share this knowledge.

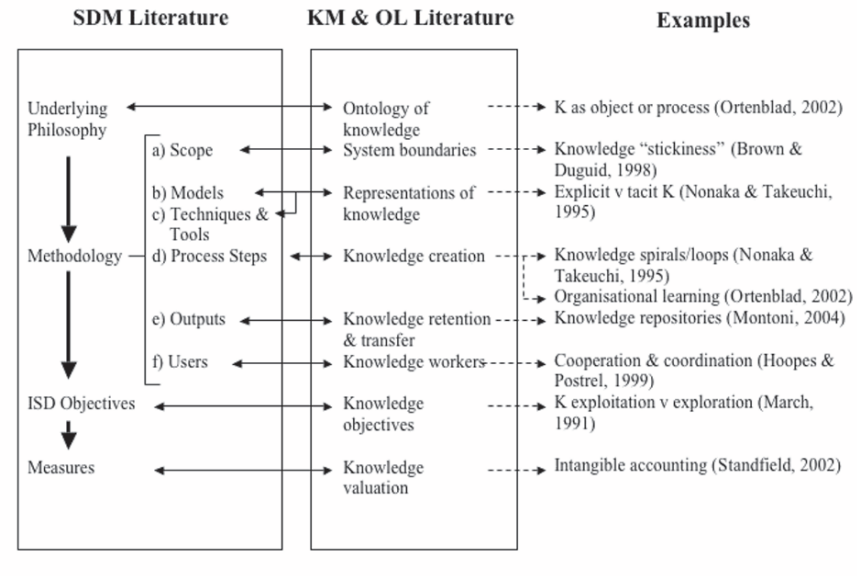

Figure 1. Mapping of KM and OL concepts onto the hierarchy of SDM elements.

The objectivist philosophy is contested by researchers who emphasize the cognitive and social aspects of knowledge and see knowledge as situated in a process of interactions (Brown and Duguid 1998). For example, the idea of communities of practice belongs to this subjectivist branch of the literature. Within these two schools there are of course numerous variations and stances.

Care must therefore be taken when combining and applying different elements of knowledge management theory in case there philosophical inconsistencies between them and the aspect of the methodology being considered. For example, during design will the emphasis be on documentation (objectivist) or workshops and walkthroughs (subjectivist)?

Different methodologies have different scopes in terms of which aspects of the systems development process that they include. This scope will determine the set and type of knowledge covered by the methodology. This can help management to judiciously determine where inter- and intra-project boundaries should be drawn; such as which user representatives should be included in the core project team. Knowledge can be "sticky" and have difficulty crossing organizational and project boundaries and so managing knowledge flow across these boundaries is an important task. There may be a need for roles responsible for translating between the different domains or to broker, mediate and coordinate the transfer of knowledge (Brown and Duguid 1998).

The way knowledge is represented is a central theme in knowledge management. The models, tools and techniques, process steps and outputs of

a methodology will determine how project knowledge is represented and therefore created, retained and transferred. Structured analysis tools, such as data flow and process flow diagrams, aim to make knowledge as explicit as possible. The story cards used in agile methodologies, on the other hand, leave much of the knowledge required to develop from them tacit. Explanations of knowledge generation and organizational learning emphasize the circulation of knowledge flows within the organization (Crossan et al. 1999; Boisot 1998; Nonaka and Takeuchi 1995), which has implications for the appropriateness of different process models as the frequency of project iterations affects how quickly a project can learn.

As for the users of methodologies, Walz (1993) found that for some projects over 75% of the time devoted to the design phase was spent in learning both the user requirements and technologies to be used. Their recommendation is that managers should increase the amount of application domain knowledge across the entire software development staff by auditing the existing knowledge of project members, allowing time for learning, and provide tools to help capture information and project experiences for later reuse.

However, learning, sharing and integration are time consuming and every member of the team does not have to know everything. Postrel (2002) suggests that given the costs of developing trans-specialist knowledge (i.e. knowledge outside of one's own specialist area) the best approach is for workers to specialize in their own knowledge domains and for management to foster "islands of shared knowledge" only where necessary. This suggests that management should be more actively involved in managing learning and the flow of knowledge than much of the ISD and product development literature proposes.

In terms of objectives, what should be the balance between knowledge exploration – the discovery and creation of new knowledge – and exploitation – the systematic and purposeful application of knowledge (March 1991)? In ISD projects there will be conflict between stakeholders because of the tensions between the working conditions conducive to thinking and creativity and the economic pressures to complete projects as quickly and cost effectively as possible. Different methodologies provide management with varying degrees of direct control over work tasks and therefore require different levels of cooperation from workers for their successful use. Although managers need to consider what individual, project and organizational learning is required to successfully develop the system that is the immediate focus of concern, it will be prudent to be mindful of what further investment should be made in learning beyond immediate project needs. This will engender competences relevant for future projects – and associated activities such as tendering.

In ISD exploration and exploitation activities often happen together but some activities, such as requirements definition and design, are more focused on exploration than others, such as coding and testing. Knowledge

exploitation in ISD is becoming progressively easier due to technology improvements in areas such as hardware, packaged software and development tools.

Similar improvements are required in knowledge exploration (Drucker 1999; Brooks 1987). The amount of knowledge exploration and therefore learning that takes place varies widely by methodology. In general iterative methodologies, such as those which incorporate a spiral process model, will encourage exploration and learning. On the other hand the more established waterfall-type models emphasize more of a knowledge exploitation approach because knowledge exploration is prescribed only in the very early phases.

However generalizations like these may be unhelpful. For example, eXtreme Programming (XP) takes the idea of multiple iterations to the extreme: ideally a project is a series of two-weekly releases. However, as Stephens and Rosenberg (2003) point out, in practice a number of XP practices are in fact anti-learning. Simple design and constant refactoring reduce the amount of reflection and thinking ahead. Pair programming may discourage an individual from working through a problem if their partner knows how to solve it. Additionally, a customer representative based on-site creates a single point of contact for all the projects external knowledge needs. XP therefore represents an extreme knowledge exploitation strategy that relies on already skilled programmers and knowledgeable customer representatives to be successful.

A final concern for managers is how performance may be measured. A knowledge based perspective emphasizes systems quality over the quantity of deliverables produced. Most of the literature on knowledge measurement and valuation is directed at the firm-level, although Standfield (2002) is creating standards for intangible accounting and management that may prove useful to ISD. The subjectivist perspective on this however is that the search for metrics is counterproductive as it attempts to reify knowledge and "such indicators do not provide any sense of an organization's stock or flow of knowledge or its contribution to decision making and organizational performance." (Fahey and Prusak 1998)

6. The Future

Fitzgerald (2000) suggests that any new methodologies should focus on simplifying and speeding up ISD by the following means:

1. Making greater use of packaged software and outsourcing, which allow systems to be developed with higher level building blocks;
2. Recognizing that most business software is algorithmically simple and therefore using methods and tools that are correspondingly straightforward;

3. Aiming for satisficing solutions that are 'good enough' for the business need rather than striving to deliver something excessively functional or sophisticated.

At the same time, methodologies should allow developers to use both SDLC (top-down) and prototyping (bottom-up) approaches to elicit requirements as needed as well as giving developers autonomy to choose their own design and implementation methods.

The above criteria, therefore, combine to change the emphasis to one that reinforces both pragmatic considerations and also that a methodology is usually there to aid rather than hinder a developer.

If future methodologies follow this line then future methodologies are likely to deal at a higher level of abstraction, specifying desired outcomes (what) rather than prescribing the exact steps to be followed (how). They should also give guidance on determining the learning needs of project staff, how requirements should be negotiated, how conflicts inherent in the creational processes can be resolved and how particular factors such as these contribute to the project's uncertainty and risk (Curtis et al. 1988).

A knowledge perspective would add to and amplify these factors by providing an assessment of knowledge needs i.e. what knowledge is available, what knowledge needs to be acquired and what knowledge will be generated; the most appropriate process model for the needs of knowledge exploration versus exploitation; the appropriate project organization and technologies for knowledge sharing, sympathetic with worker cooperation and knowledge coordination. It may be that methodologies based on the conception of ISD as an engineering discipline may become less prevalent and a new perspective and set of corresponding methods will come to the fore.

As an example of how knowledge-based thinking can contribute to a better understanding of the future of methodologies, Boisot's (1998) I-space provides a strategic model for analyzing organizations from a knowledge-based perspective. He proposes that organizational cultures can be characterized as fiefs, bureaucracies, markets and clans, which in the simplified I*-space version of his model occupy each of the four quadrants of a 2x2 matrix mapping knowledge codification against knowledge diffusion.

Boisot's bureaucracies and markets in the top half of I*-space correspond to common business usage of these terms. In bureaucracies knowledge is highly documented (codified) to allow knowledge retention and sharing, but the diffusion of this knowledge is strictly controlled within the organizational hierarchy by management. In markets knowledge is also highly codified but sharing is uncontrolled and therefore information is widely diffused. Business relationships are impersonal and competitive and coordination is through processes of mutual adjustment and self-regulation. In the bottom half of I*-space, fiefs are small organizations, such as business

start-ups, where knowledge is largely uncodified and undocumented. Relationships are face-to-face and hierarchical. Members of a fief are expected to have shared beliefs and goals and to subordinate themselves to the goals set by their leader. Finally, clans are typified by business or academic networks. The valuable knowledge in these networks is largely uncodified and passed through personal contact. A clan's goals and activities are negotiated by its members who must therefore share common values and beliefs.

Applying Boisot's I*-space to the process of information systems design provides insights into the choice of systems development methodologies. Historically for large projects, IT managers chose between in-house development using traditional, formalized methodologies and using third-party solutions either in the form of packaged software or by outsourcing software development. The newer, agile methodologies described above along with the use of open source software have been receiving increasing interest as possible software development methods. These four development approaches – formalized methodologies, packaged software and outsourcing, agile methodologies and open source software development – map onto Boisot's four organizational cultures in I*-space as shown in Figure 2.

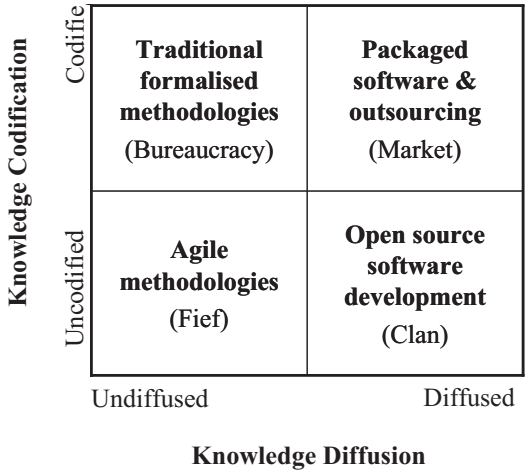

Figure 2. ISD approaches mapped onto Boisot's I*-Space.

Information technology influences organizational form. Improvements in information technology are constantly increasing the bandwidth with which information systems can process and transmit information. This increase allows organizations either to transmit more information, more quickly and more widely, increasing the diffusion of knowledge; or to reduce the amount of codification that is required prior to transmission. In consideration of these changes, Boisot hypothesizes that there will be a shift in organizational forms from bureaucracies in the top left of I*-Space to the right (greater

diffusion) and down (less codification) towards clans in the bottom right quadrant.

The emergence of agile methodologies and open source software development as viable options for software development can be attributed to the improvements in information technology hypothesized by Boisot. In the case of agile methodologies, this stems from a combination of improved modeling and programming techniques (particularly object oriented techniques) (Glass 1999) and increased machine speeds allowing fast compilation and testing. The accessibility of open source software development results from the use of the internet as a communication medium (Raymond 1997).

Organizations will continue to take positions in I*-space but this analysis suggests that there will be a move from formalized methodologies to approaches incorporating the principles of open source development (as shown by the arrow in Figure 3 above). The bottom right quadrant in I*-space (clan/open source software development) is also the culture that is suggested by much of the subjectivist literature of organizational learning (Ortenblad 2002).

7. Application to Current Practice

New knowledge-based methodologies will take time to emerge. Our research indicates that the selection of methodologies is determined as much by stakeholder preferences – which are usually towards simple to understand waterfall approaches – as the characteristics of the projects themselves. The IT departments are therefore constrained in the methodologies that they can use and see part of their role as educating the business about possible approaches.

This does not mean that knowledge management ideas cannot be usefully applied to existing practice. Recent case study research by one of the authors investigated the management of project issues during ISD using traditional methodologies. The root causes of many of the issues were found to be knowledge gaps (Hoopes and Postrel 1999): the knowledge needed to successfully perform the task existed within the project but this knowledge was not effectively used due to problems of knowledge sharing. Many knowledge sharing problems during requirements specification, systems analysis and systems design were caused by poor working relationships between the designers and other project participants.

The results of this research were used to develop a workbook – comprising methodologies to audit project performance and improve project control – to help project managers anticipate and manage the risks of knowledge gaps during the design of Information Systems. Central to this workbook was a mapping of the phases of the systems development lifecycle to Boisot's four organizational cultures in I*-space as shown in Figure 3. This mapping provided an "ideal" case for the project organization and control against which to evaluate the particular project. For example, based

on knowledge management theory, systems analysis and systems design ideally involves a fief culture involving a small, cohesive team closely supervised by the project manager. In practice the design team often involves participants from disparate organizations such as the IT department, the customer and third party suppliers and these participants may have little loyalty to the project manager or one another. The project manager therefore should be acutely aware of the risk of poor cooperation between the various participants and put in place the appropriate formal and informal control modes to manage their work, as well as contingencies to deal with any residual risks.

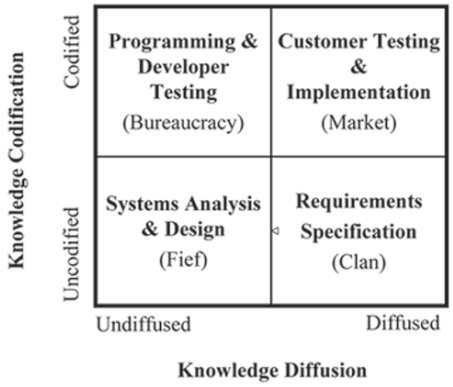

Figure 3. SDLC phases mapped onto Boisot's I*-Space.

8. Next Steps and Conclusion

This paper has presented an overview of the history, theory and state of practice of systems development methodologies. It has also presented some ideas from the knowledge management literature and proposed that these should be incorporated into the future development of methodologies and so provide greater structure for the coordination of activities within ISD projects. Given the different viewpoints about the nature of knowledge and the role of management, it is certain that – as with existing methodologies – knowledge-based methodologies will be developed that reflect both different philosophies, types of projects and stages of information systems development.

We are pursuing two streams of work. The first involves gathering empirical evidence for the value of the knowledge perspective in practice. Practice has often preceded theory in the field (Fitzgerald 2000) and research is being directed at examining whether knowledge management and organizational learning ideas are in fact influencing ISD practice. The second stream is developing knowledge-based methodologies following the

guidelines presented in this paper with the intention of using them in action research.

References

Abrahamsson, P: 2002, *Agile Software Development Methods. Review and Analysis,* VTT Electronics, Oulu, Finland.

Avison, DE and Fitzgerald, G: 2003, Where now for development methodologies?, *Communications of the ACM* **46**(1): 79-82.

Avison, DE and Wood-Harper, AT: 1991, Information systems development research: An exploration of ideas in practice, *The Computer Journal* **34**(2): 98-112.

Barry, C and Lang, M: 2002, A comparison of 'traditional' and multimedia information systems development practices, *Information and Software Technology* **45**(4): 217-227.

Boehm, B and Turner, R: 2003, *Balancing Agility and Discipline: A Guide for the Perplexed,* Addison-Wesley, Boston.

Boisot, MH: 1998, *Knowledge Assets: Securing Competitive Advantage in the Information Economy,* Oxford University Press, Oxford.

Brooks, FP: 1987, No silver bullet: Essence and accidents of software engineering, *Computer* **20**(4): 10-19.

Brown, JS and Duguid, P: 1998, Organizing knowledge, *California Management Review* **40**(3): 90-111.

Crossan, MM, Lane, HW, White, RE: 1999, An organizational learning framework: From intuition to institution, *Academy Of Management Review* **24**(3): 522-537.

Curtis, B, Krasner, H and Iscoe, N: 1988, A field study of the software design process for large systems, *Communications of the ACM* **31**(11): 1268-1287.

Drucker, PF: 1999, Knowledge-worker productivity: The biggest challenge, *California Management Review* **41**(2): 79-94.

Fahey, L and Prusak, L: 1998, The eleven deadliest sins of knowledge management, *California Management Review* **40**(3): 265 - 276.

Fitzgerald, B: 2000, Systems development methodologies: The problem of tenses, *Information Technology and People* **13**(3): 174-185.

Glass, R: 1999, The realities of software technology payoffs, *Communications of the ACM* **42**(2): 74-79.

Hoopes, DG and Postrel, S: 1999, Shared knowledge, "glitches," and product development performance, *Strategic Management Journal* **20**(9): 837 - 865.

Kettunen, P: 2003, Managing embedded software project team knowledge, *IEE Proceedings - Software* **150**(6): 359-366.

March, JG: 1991, Exploration and exploitation in organizational learning, *Organization Science* **2**(1): 71-87.

Montoni, M, Miranda, R, Rocha, A and Travassos, G (eds): 2004, *Knowledge Acquisition and Communities of Practice: An Approach to Convert Individual Knowledge into Multi-Organizational Knowledge*, Lecture Notes in Computer Science 3096, Springer.

Nonaka, I and Takeuchi, H: 1995, *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation,* Oxford University Press, New York.

Ortenblad, A: 2002, Organizational learning: A radical perspective, *International Journal of Management Reviews* **4**(1): 87-100.

Postrel, S: 2002, Islands of shared knowledge: Specialization and mutual understanding in problem-solving teams, *Organization Science* **13**(3): 303-320.

Raymond, ES: 1997, *The Cathedral and the Bazaar*, Available Online, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Riemenschneider, CK: 2002, Explaining software developer acceptance of methodologies: A comparison of five theoretical models, *IEEE Transactions on Software Engineering* **28**(12): 1135-1145.

Robillard, PN: 1999, The role of knowledge in software development, *Communications of the ACM* **42**(1): 87-92.

Standfield, K: 2002, *Intangible Management: Tools for Solving the Accounting and Management Crisis,* Academic Press, San Diego.

Stephens, M and Rosenberg, D: 2003, *Extreme Programming Refactored: The Case Against XP,* Apress, Berkeley, CA.

Walz, DB: 1993, Inside a software design team: Knowledge acquisition, sharing and integration, *Communications of the ACM* **36**(10): 63-77.

Wastell, DG: 1996, The fetish of technique: Methodology as a social defence, *Information Systems Journal* **6**(1): 25-40.