

ATHABASCA UNIVERSITY

A SYMBIOSIS BETWEEN AGILE METHODS AND KNOWLEDGE  
MANAGEMENT FOR DEALING WITH COMPLEXITY IN SOFTWARE  
ENGINEERING

BY

JOHN W. SCOTT

A thesis submitted in partial fulfillment  
Of the requirements for the degree of  
MASTER OF SCIENCE in INFORMATION SYSTEMS

---

Athabasca, Alberta

April, 2006

© John W. Scott, 2006

## APPROVALS

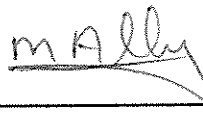
The undersigned certify that they have read and recommend for acceptance the thesis A SYMBIOSIS BETWEEN AGILE METHODS AND KNOWLEDGE MANAGEMENT FOR DEALING WITH COMPLEXITY IN SOFTWARE ENGINEERING submitted by JOHN WILLIAM SCOTT in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in INFORMATION SYSTEMS.



Fuhua Lin, Ph.D.  
Supervisor



Steve Leung, M. Sc IT  
Reviewer



Mohamed Ally, Ph.D.  
Chair

Date: May 18, 2006

## **DEDICATION**

Throughout the course of my studies I have had the encouragement and support of many people. First among them my mother, Bernice, who has only asked that I apply myself whole heartedly and done my best.

## **ABSTRACT**

This thesis proceeds from the notion that complexity in the field of software engineering is a result of the internal and external forces that must be brought together to engage in it, resulting in a chaotic state that breeds failure and that this state can be ameliorated through the use of the Agile Methods and knowledge management. After enumerating and discussing the sources of complexity, and surveying the results of their interactions, it proceeds to look at the theory of knowledge management. In elaborating the ideas of Nonaka and Takeuchi we are introduced to the concepts of both the knowledge spiral and a model of knowledge creation, including its enabling conditions and barriers, as well as the concept of Ba. These theories seek to harness the interplay of explicit and tacit knowledge to manage complexity. These theories form a framework that is then applied to the Agile methods, specifically Extreme Programming and the Crystal Clear methodologies, in order to evaluate their values, principles, practices and strategies of software engineering from a knowledge management perspective. This evaluation finds that there is a congruency between the practices of the Agile methods and the theory of knowledge management. In particular the manner in which the Agile methods engage in an iterative pattern of development, its support for direct communication, the incorporation of non-technical team members in defining the product and planning and its reflection on process and product, create a knowledge creating process. As a result, many of the issues of complexity are either removed or minimized through the application of this process.

## **ACKNOWLEDGEMENTS**

Few meaningful contributions come from the efforts of one person acting alone. Throughout this degree I have had the opportunity and pleasure of interacting with many Professors and fellow students. Each of them has added something to my understanding and vision of the field. For that I am thankful.

Dr. Fuhua Lin, my supervisor while I wrote this thesis, and several courses which drove me in the direction of it, I thank for his encouragement, comments and willingness to allow me to find my own path, while nudging me in the right direction.

I would also like to thank Dr. Mohamed Ally who chaired the review committee for this thesis and Steven Leung who acted as reviewer.

## TABLE OF CONTENTS

DEDICATION.....	III
ABSTRACT.....	IV
ACKNOWLEDGEMENTS .....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES.....	VIII
LIST OF FIGURES .....	IX
CHAPTER I - INTRODUCTION .....	1
Introduction.....	1
Statement of Purpose.....	2
Significance .....	3
Limitations .....	3
Conduct of Research.....	4
Organization of the Essay .....	5
CHAPTER II - COMPLEXITY IN THE SOFTWARE ENGINEERING CONTEXT.....	7
Introduction.....	7
Sources of Complexity.....	7
Complexity and its Outcomes .....	14
Conclusion.....	16
CHAPTER III - KNOWLEDGE MANAGEMENT .....	18
Introduction.....	18
The Knowledge Continuum .....	18
Types of Knowledge .....	21
Knowledge Creation .....	23
Knowledge Spiral .....	24

Knowledge Enablers and Barriers .....	29
BA.....	38
Software Engineering and Knowledge Management.....	40
Software Engineering and Knowledge Management Technology .....	44
Conclusion.....	47
CHAPTER IV - AGILE METHODS AND KNOWLEDGE MANAGEMENT .....	48
Introduction.....	48
The Agile Movement.....	48
Knowledge Creation in the Agile Methods.....	50
Agile Methods: Enablers, Barriers and Ba.....	59
Agile Lifecycles and the Knowledge Creation Process.....	68
Conclusion.....	70
CHAPTER V - CONCLUSIONS AND RECOMMENDATIONS .....	72
Introduction.....	72
Addressing Software Complexity.....	72
Recommendations.....	76
REFERENCES .....	78
APPENDIX A .....	87

## LIST OF TABLES

Table 1 Compilation of Data from the Standish Chaos Reports.....	15
Table 2 Factors in Project Success/Failure.....	15



## LIST OF FIGURES

1. Spiral Model of the Software Process .....	10
2. Spiral of Knowledge Creation .....	25
3. Five-phase Model of the Organizational Knowledge-creation Process .....	28
4. Conceptual Representation of Ba .....	39
5. A Dynamic Model of Software Engineering Knowledge Creation .....	42
6. Extreme Programming Project Model .....	69
7. Crystal Clear Project Model .....	69

## CHAPTER I

### INTRODUCTION

Rather than seeking an unachievable stability, software organizations should focus on creating software engineering knowledge and mind-sets that embrace environmental change.

(Dybå, 2003)

#### Introduction

The venue in which software construction takes place is fraught with complexity. This complexity derives from multiple sources, both internal to the software engineering process and external to it in the form of the demands placed upon it by customers, schedules and change.

Knowledge management seeks to improve the flow of data, information and ultimately knowledge between individuals in order to create an organization that can better deal not only with the complexity of the knowledge it currently has, but to enable it to create new knowledge and innovation from this established base, allowing it to take effective and timely action in line with its goals. While knowledge management has for the most part been applied to the larger organizational context, it seems no less applicable to a project based team engaged in software construction where data and information must be acquired from diverse sources and embodied in a system that represents certain knowledge of the businesses environment, processes and goals.

To achieve a higher degree of agility, a software engineering paradigm must take into account the environment in which it exists and strive to improve its

ability to acquire and utilize knowledge from both its own processes and from those who have invested in it as a solution to their needs. It is within the symbiosis between knowledge management and the Agile methods that we can discover this agility. Agile methods represent an evolution of previous software engineering practices. Stepping back from overly hierarchical and document driven approaches, it stresses the importance of the individual, communication and adaptability as central themes in creating software.

### Statement of Purpose

This thesis will look at the root causes of complexity within software engineering and the methods by which practitioners and academia have advanced to overcome them. The author proposes to evaluate the potential of knowledge management to assist in dealing with complexity in software engineering, which is fundamentally as a problem of the acquisition and dissemination of project knowledge. Particular attention will be focused on the use of Agile methods in dealing with certain aspects of complexity. In particular:

- Identify those aspects of Agile methods which represent opportunities to acquire project data, information and knowledge
- Evaluate current principles, practices and values within the Agile methods used to capture and make use of such collected project data, information and knowledge
- Review these principles, practices and values with reference to the principles of knowledge management and evaluate their completeness and effectiveness in dealing with complexity

- Proposing and extending principles, practices and values within the Agile methods to improve the capture and dissemination of project knowledge. and finally
- Maintain the spirit of Agile methods.

### Significance

A significant amount of research has been conducted in the area of knowledge management and software engineering. Much of this work has been carried out with a view to improving what can be termed infrastructure, ancillary tools which promise to assist with software development. While this is important it tends to overlook the vital contribution of individuals to what is at its core a creative and innovative activity. Software engineering, as a knowledge intensive process, is likely irreducible to a strictly process oriented framework and will continue to rely on the creative efforts of people. Through the elaboration of a symbiosis between Agile methods and knowledge management our ability to deal with software complexity can be improved by focusing our attention on the creation of project knowledge and insuring its dissemination throughout both the project team and the customer community.

---

### Limitations

Although the particular choice of platform, system architecture or language used to construct a system can have entail own complexities, this thesis does not address them in any detail. In a given situation skilled software engineers can make appropriate choices in these areas. While the solution to software engineering complexity may someday be found within these areas the author

believes that it is the surrounding processes that cause the majority of difficulties today and therefore offer the most promising areas for breakthroughs and improvement.

The issues surrounding the enabling of distributed Agile development and the applicability of Agile methods in large project development settings are beyond the scope of this enquiry, as the knowledge management aspects of it need to be addressed before trying to distribute or scale the methodologies.

### Conduct of Research

The idea for this thesis originated in the readings and ideas that the author has been exposed to over the course of his studies. Having been introduced to the concepts of knowledge and knowledge bases, software engineering and Java programming based on test driven development, and my own interest in XP, the author became interested in how these areas interact. Of particular interest were the problems that seem to plague software engineering and its ability to achieve success. After looking into the field of knowledge management further, parallels became apparent between the cycle of knowledge creation and innovative processes and the cyclical nature of many of the existing software development paradigms, which can be seen as a method of elaboration and reduction to a specific understanding of a particular problem into a coded system. The thesis from this point took on three dimensions; the complexity of software engineering and its consequences, the import of knowledge management in dealing with complexity and the Agile methods as a potential response.

Research began with extensive searches of literature databases, including the ACM (Association for Computing Machinery), IEEE Computer Society Digital Library, Elsevier's Science Direct, Blackwell Synergy, Springer Link and Athabasca's Electronic Journals database which yielded results from the Harvard Business Review and Sloan Management Review among others. Google searches were also conducted and resulted in the identification of sites and materials which were of interest for all of the components of the thesis. From this initial set of resources, leading articles and authors were identified and read to better understand the areas they covered. Additional resources were then found through a review of the sources they cited, and updated where necessary, to identify the most relevant and current.

#### Organization of the Essay

The thesis is divided into several sections in order to examine and then bring together the areas of knowledge management and Agile methods as a possible solution to the complexity of producing software.

Chapter I – This chapter, is an introduction to the thesis describing in general the purpose, aims and manner of conducting of it.

Chapter II – Complexity in the Software Engineering Context, begins this process by reviewing the literature related to software engineering complexity. By identifying those areas within the software engineering process which cause the greatest concern and looking at the results of several studies into the causes of software failure, we can identify those areas in need of a new way of dealing with them.

Chapter III – Knowledge Management, looks into the origins and major tenants of knowledge management which will serve to underpin the investigation into its applicability to software engineering.

Chapter IV -- Agile Methods and Knowledge Management, begins by elaborating the precepts of the Agile movement. It will then apply the ideas canvassed in the previous chapters and evaluates the degree to which Agile methods currently embody aspects of knowledge management and the potential for further gains through a more encompassing use of them in dealing with complexity.

Chapter V – Conclusions and Recommendations, seeks to find the balance between knowledge management and Agile methods as well as offering several areas which may merit further investigation.

## CHAPTER II

### COMPLEXITY IN THE SOFTWARE ENGINEERING CONTEXT

#### Introduction

Software engineering is composed of many divergent fields all of which must be brought together to act in concert to successfully complete a project. Each of these disciplines has their own complexities which are compounded by their integration into a whole. At the same time software engineering must remain cognizant of and function within the wider context of the sponsoring organization, adding yet another layer of complexity which must be managed if a project is to be successful.

#### Sources of Complexity

The sources of complexity in software engineering are both internal and external to its process. What follows is a brief description of these sources of complexity and the impacts they can have on project viability.

#### Internal

The Software Engineering Body of Knowledge (SWEBOK) (Abran, Moore, Bourque, & Dupuis, 2004) makes reference to 10 primary knowledge areas and 8 related disciplines that form the body of knowledge for software engineering. While all of the enumerated areas and disciplines are important, those that are of specific interest to this thesis from the knowledge areas include: software requirements, software design, software engineering process, specifically, software life cycle models, software testing and software maintenance. From the



SWEBOK's related disciplines the area of project management is of interest as well. In addition to the areas just identified (Schach, 2002) also includes the composition and organization of the team undertaking the project as an important consideration.

### Software Requirements and Change

The elicitation of customer requirements has traditionally been carried out in the initial stages of a projects life. This makes logical sense as, in order to design and plan a project we need to know what features and functionality the project will be expected to deliver. What it does not take into account are the inevitable changes to the requirements, necessitated by shifts in the customers understanding of their needs as the system takes shape allowing them to better understand and assess its potential and their desire to take advantage of them. Changes in requirements after the initial stages of analysis and design can have significant impacts on the cost and schedule of a project and can even result in having to revisit the design of the systems architecture if they represent a radical departure from what was originally anticipated. While various project management practices attempt to deal with uncertainty in software development through the application of critical path analysis, program evaluation and review technique (PERT) and risk management through risk/impact assessments, pose a significant threat if they are not managed properly (Verner & Evancho, 2003). The main method of mediating the effects of change in the projects initial requirements comes from change control management, which while offering the

opportunity to shift amongst the triumvirate of scope, time and cost (Schwalbe, 2002) does little to address the root causes of the need to do so.

### Software Architecture and Design

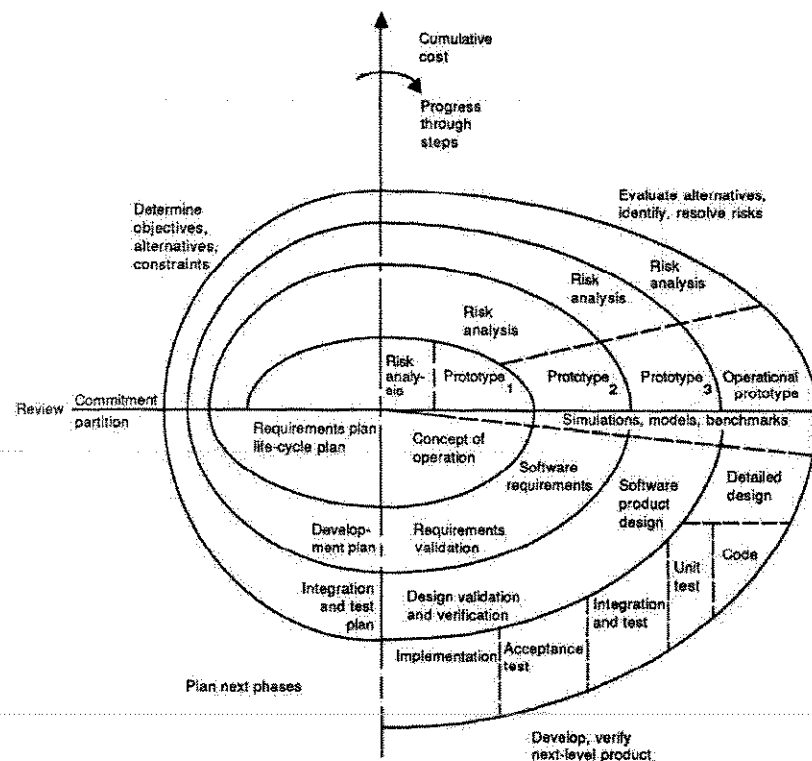
The architecture and detailed design phase of software development are in and of themselves complex, but can also impact on the projects ability to accommodate change during construction and over useful the lifespan of the software once it has entered the maintenance stage. A systems' architecture must not only provide the foundation for the system under construction, but allow enough flexibility to accommodate integration with existing systems and also for future enhancement. Detailed design serves to further lock in the features that a system will support, while at the same time closing the door on other avenues of modification.

### Software Life Cycle Models

Many paradigms of software engineering have been advanced. Among the more accepted traditional ones is the waterfall approach advanced by (Royce, 1987) which proceeds in an orderly fashion from an initial set of system and software requirements through analysis, design, coding, testing and concludes when the software has entered service. While this model offers some improvement over what has been termed the "code and fix approach" (Boehm, 1988), describes the process as basically writing some code, while being aware of the requirements, design, need for testing and maintenance, but paying little heed to them. The waterfall approach at least established a rough order in which some consideration is given to the utility and implementation of the software

before coding begins. On the other hand by segmenting the development process into a series of sequential stages the waterfall model makes it very difficult and costly to revisit a prior stage should it be determined that any of its assumptions was incorrect. Boehm's contribution to the software process model genus is the spiral model as shown in Figure 1 below.

*Figure 1. Spiral Model of the Software Process*



From "A Spiral Model of Software Development and Enhancement.", Barry Boehm, IEEE Computer, 21(5), Copyright © 1988 IEEE. Reprinted with permission.

The spiral model improves on the waterfall model through the continual re-visitation of the various stages of the development process. Beginning each

phase of the spiral with careful consideration of the objectives, alternatives and constraints for that iteration, as well as what has been learned from the previous phase/cycle, an assessment of risk can be made so as to inform the next phase/cycle. The cycles continue to refine the prototype until no further additions or changes to the prototype are required (Boehm, 1988). While the spiral model incorporates an important element of risk management, it still proceeds from a fixed set of requirements and moves through stages not unlike the waterfall model to completion albeit in an iterative manner. Other models have been suggested, such as Rapid Application Development (J. Martin, 1991), the Rational Unified Process (Kroll & Kruchten, 2003), but continue to behave more or less in the same manner, trying to lock down requirements prior to the analysis and design phases, making the accommodation of change beyond this “requirement freeze” stage difficult.

### Software Testing

Within the life cycle models just enumerated concerns can be raised with the late point during the development cycle at which testing is undertaken. Naturally testing cannot occur until some requirements have been elicited, design and coding have begun, but to lump unit, integration and acceptance testing to the latter stage of development gathers the risk of difficulties to a single point of failure rather than spreading it out and allowing more time to detect and correct such flaws.

### Software Maintenance

It has been shown that expenditures (and hence time) constitute in the area of 67% (Schach, 2002) of the cost of software. The need therefore to make software maintainable from its earliest stages through its design, implementation and documentation adds another degree of complexity. This complexity arises from the need to foresee or predict what the software might be required to do in the future.

### External

#### Project Sponsorship

The inclusion of key stakeholders, whose knowledge of the existing practices and processes in the area under consideration, as well as their understanding of the politics of the organization, can also guide in determining what will have to be accomplished not only to produce the product but to assure its acceptance.

#### Leadership Knowledge of Business Need

A project must address a particular business need. Poorly articulated organizational goals that form the basis of a project will pose significant and most likely fatal difficulties for it. Without clear goals the project will lack direction and in short order will also lose the customers' attention, support and interest. If the project survives and is ultimately deployed in such a situation it is unlikely that it will serve any useful purpose. Rapidly shifting business environments necessitating change to established business processes, models and needs as

well as legal compliance issues, will filter down and cause changes in project requirements.

### IT and Business Need

A study conducted by (ATKearney, 2005) found that business views IT as primarily concerned with day to day operations, less willing to be early adopters of new technologies and less likely to propose innovative solutions that could yield competitive advantage to the organization. Overall the sense is that IT is not a proactive part of the organization, but rather a reactive entity, and a hesitant one at that. This should not come as much of a surprise as the same study points out that the majority of IT budgeting is focused on maintaining the current systems while only 20% is spent on innovative initiatives. The remedy, according to the study, is to determine how each aspect of IT benefits the business, reduce complexity and standardize infrastructure. The resulting assessment will identify those areas of IT which are essentially commodities. These commoditized areas, in the sense that (Carr, 2003) proposes, while necessary are not likely to offer any competitive advantage to the organization due to the ease and speed with which they can be adopted by competitors. Such operations should be outsourced to allow IT to focus on areas which have the potential to add value and innovative practices (ATKearney, 2005).

### Customer Involvement

While it is essential to have a committed project sponsor and a clear business need for a project to gain traction, the involvement of internal line-of-business customers at the early stages of requirements definition and throughout

the development process is key to ensuring that the product being defined integrates their vision for it.

Customers are not generally software engineers and we therefore cannot expect them to have the same level of understanding of the place and potential of information technology within their organization, but they do tend to know what they need in order to fulfill their functions. IT needs to develop an informative environment, to relate its relevance, possibilities and purpose in language that the customer understands.

#### Complexity and its Outcomes

How then does IT fair in its attempts to deliver solutions. Ultimately the success or failure of a particular project must be measured by its ability to meet the objectives it was initiated to address. It is within the analyses of project success and failure that we can discern the effects of complexity and software engineering's ability or inability to deal with them, The Standish Group (The Standish Group Report - CHAOS, 1995) uses the following categorizations to classify the outcomes of the projects they studied. Projects are categorized as successful when they are on time, on budget and deliver the functionality originally specified. Projects are considered challenged, if they are delivered over budget and beyond their expected schedule with reduced functionality. Failed (noted as impaired in some reports) projects are those that are cancelled at some point during their development. A compilation of the results from the bi-annual reports is shown below in Table 1.

Table 1 Compilation of Data from the Standish Chaos Reports

Year	Success	Challenged	Failed
1994	16.2	52.7	31.1
1996	27	33	40
1998	26	46	28
2000	28	49	23
2002	34	51	15
2004	29	53	18

Note. All figures represent percentages. Adapted from: (CHAOS: A recipe for success, 1999; Glass, 2005; iTWire, 2004; The Standish Group Report - CHAOS, 1995)

The reports go on to specify various factors that lead to the enumerated outcomes. The factors cited for project success and failure in (The Standish Group Report - CHAOS, 1995) report are shown in Table 2.

Table 2 Factors in Project Success/Failure

Rank	Successful	Challenged	Impaired
1	User Involvement	Lack of User Input	Incomplete Requirements
2	Executive Management Support	Incomplete Requirements & Specifications	Lack of User Involvement
3	Clear Statement of Requirements	Change Requirements & Specifications	Lack of Resources
4	Proper Planning	Lack of Executive Support	Unrealistic Expectations
5	Realistic Expectations	Technology Incompetence	Lack of Executive Support
6	Smaller Project Milestones	Lack of Resources	Change Requirements & Specifications
7	Competent Staff	Unrealistic Expectations	Lack of Planning
8	Ownership	Unclear Objectives	Didn't Need It Any



			Longer
9	Clear Vision & Objectives	Unrealistic Time Frames	Lack of IT Management
10	Hard Working, Focused Staff	New Technology	Technology Illiteracy

These categorizations and factors have not gone unchallenged as (Glass, 2005) points out the determination of success or failure are highly subjective and further that studies and investigators tend to focus on projects that are in trouble before the study began. In addressing such concerns we can look at (Boehm, 2000), where he argues that a 31% failure rate is not that surprising, and may in fact represent a realistic attrition rate for software projects, but goes on to explain that the results for “impaired” projects may be misleading in that they are not entirely due to poor practices in software engineering. Among the reasons he gives are the shifting needs and priorities of the organization and the value that a given project will provide upon its completion which could justify its cancellation, the pressure of scope change without corresponding changes in budget and time allotment, customer buy-in to the aims of the project and the rapidly changing environment.

### Conclusion

The issue of complexity in software engineering is clearly a multifaceted one. As stated by (J. Martin & McClure, 1988) “the basic problem of computing is the mastery of complexity”, in fact the problem goes far beyond the limited area of computing and in fact extends into every aspect of software engineering. What is needed is a more holistic view and understanding of the place of software

engineering within its larger context and a more reactive and innovative nature. Complexity in software engineering has heretofore been dealt with as a patchwork of responses, efforts to bring the customer into the process with requirements engineering, the use of abstraction in the design and construction of software, and various project management practices in relation to its planning provide examples of these efforts. While this has allowed us to manage complexity it has not allowed us to master it as the studies noted clearly show. In my view complexity cannot exist where knowledge informs us. In the next sections we will look at the potential of knowledge management to provide such insights.

## CHAPTER III

### KNOWLEDGE MANAGEMENT

#### Introduction

In this section of the thesis we will investigate the concept of knowledge management. After defining the constituent elements of knowledge, we will move on to look at knowledge creation as both a method of producing new and innovative solutions and in its ability to assist us in mastering the complexity and chaos inherent in any situation of change. Finally, the efforts that have been made to infuse software engineering with elements of knowledge management will be reviewed. Much work has been done in the area in order to provide what the author terms a “knowledge infrastructure”, that is to say, technological solutions intended to overcome the complexity outlined in the second chapter of this thesis.

#### The Knowledge Continuum

The knowledge continuum exists in the interplay of data, information and knowledge and the ability of individuals to transform one form to the next. It is in the application of our mental processes that data becomes information and information becomes knowledge. One would hope that the linearity of this continuum would exist as only an upward trend (data to information to knowledge), but this has not proven to be true as (Davenport & Prusak, 1998) point out that the proliferation of knowledge leads to a loss of relevance when the individual is overwhelmed by its sheer volume reducing it to lesser forms. What

then are data, information and knowledge? And how are they transformed from one form to another?

### Data

Data according to (Tiwana, 2002) can be seen as the discreet representation of some quantifiable measurement. Data, lacking further refinement and in particular the perspective of the context in which it was collected or produced, even when given the illusion of accuracy or meaning by its quantity, contains no intrinsic meaning in and of itself.

### Information

(Drucker, 1998) tells us that "information is data endowed with relevance and purpose". Data becomes information, according to (Davenport & Prusak, 1998) when it is imbued with meaning through: contextualization, categorization, calculation, correction or condensation. This is to say that raw data is transformed, by an individual in light of their understanding and knowledge of its context and its immediate relevance to the purpose for which it is being assessed, into a message that conveys one or more of these sense-making activities.

### Knowledge

Data and information are precursors to knowledge. Information becomes knowledge in much the same way as information derives from data. In order for this to occur according to (Davenport & Prusak, 1998) individuals must assess the information on the basis of: comparisons, consequences, connections and conversations. Knowledge exhibits a specific set of attributes which delineate it

from data and information. These attributes exist and interact in an extremely fluid manner within both the individual who applies them to the information they encounter and in the knowledge that they produce. The following five attributes play a key role in defining knowledge.

#### Actionable

Knowledge is actionable in that both its means and ends find their origin in the resolution of some problem or the making of a decision. Being informed by knowledge of the situation and the various factors at play in a given context, the actions we decide to take are more likely to result in the desired outcome (Davenport & Prusak, 1998; Takeuchi & Nonaka, 2004).

#### Experience

Knowledge derives from the experiences that an individual has had in both the subjects they have studied and those situations they have been exposed to. The value of our past experiences is in the ability it gives us to extract from it a sense of what outcomes can be expected in current situations. As such experience acts as a means to deal with a new situation by granting us a perspective from which we are able to perform this evaluation (Davenport & Prusak, 1998).

#### Judgment

Knowledge of a problem space or field provides an individual with a referential basis with which they can evaluate and judge not only the problem to be solved, but also the range of solutions to be considered.

### Values / Beliefs

Knowledge contains elements of both belief and values. These measures are intrinsic to the individual who holds them and guide their evaluation of not only the content of the information they receive to but their valuation of its meaning and relevance to them in the context in which they find themselves. Knowledge once obtained becomes a “justified true belief” according to (Takeuchi & Nonaka, 2004) in the mind of the individual and to a large degree the group and organization of which they are a part.

### Complexity

Knowledge which forms the basis of our decision making ability, according to (Davenport & Prusak, 1998) exhibits the ability to deal with complexity and uncertainty precisely because of its constituent attributes, the fluidity of their interaction and the difficulty of reconciling them with one and other in a given situation.

### Types of Knowledge

#### Explicit

Explicit knowledge is knowledge that is easily reducible to some readily transmissible form such as words, documents, specifications and formulas.

#### Tacit

Tacit knowledge is much more difficult to express outside of the individuals own experience and understanding of it. This is true because the effective transmission of tacit knowledge requires not only the transfer of specific knowledge, but also the transmission of the distinctly individuated attributes that

are invested in the knowledge itself. (Nonaka, 1998) describes such knowledge as being deeply rooted in action and context, akin to the particular skills of a crafts-person, fusing a set of technical skills and a cognitive dimension found in the mental models, beliefs and perspectives that have become part of not only their knowledge, but their being. It is the sort of knowledge that requires both a formal training to transmit the technical skills, and some form of apprenticeship to begin the transfer and development of the cognitive dimension.

Tacit knowledge can be broken down into four further divisions, as (Quinn, Anderson, & Finkelstein, 1998) describe them: (1) "Know What" describes knowledge that forms a foundational level of understanding that an individual requires in order to be able to function within a particular field. For instance a programmer would have little success in their chosen field if they had no knowledge of some programming language and the algorithms that can be created with it to effect programmatic solutions. (2) "Know How" improves over know-what by demonstrating the practical application of knowledge to problems. Carrying forward our programmer analogy this could be evidenced in the selection of one algorithm over another, where either would achieve the desired result, on the basis of one being more efficient in actual use in the given situation. (3) "Know Why" again moves us up the knowledge value ladder in that one who possesses this level of knowledge are involved in more than the solution of immediate problems, but exhibit an understanding of the system and discipline as a whole, allowing them to anticipate the effects of their actions beyond the immediate concern. In the programmer this can be seen in their

consideration of the consequences of implementing a particular solution that would cause a disruption of processes outside the module they are working on or even down the road once the program entered the maintenance phase of its life-cycle. (4) "Care Why" is described as a cultural value which transcends the previous three categories of knowledge. Within it the professional is a self-motivated and creative individual who is concerned not only with their performance at the previous three levels but with keeping on top of industry trends in their field and renewing/adapting their own knowledge of the field in light of developments within it. The programmer like other professionals needs to not only keep up with the fast changing technologies and methodologies, but seek out opportunities to develop their creative potential as well.

#### Knowledge Creation

The boundary between explicit and tacit knowledge is not as impermeable as it seems. In fact (Takeuchi & Nonaka, 2004) state that they are in fact interdependent and interpenetrating in that the existence of one is dependent on the other. This symbiotic relationship is in fact critical to the knowledge creation process. Of particular note with respect to knowledge creation is the dichotomy between eastern and western systems of scientific philosophy that some of the authors (Takeuchi & Nonaka, 2004) point out. The West they offer has adopted the Cartesian or scientific model of knowledge, relying on hard facts, mathematical and statistical proofs to the exclusion of the more Eastern model which admits of a more holistic set of evidences encompassing the individuals' feelings and intuitions regarding the subject of enquiry. As such the Western



tradition is seen to exclude that which is not reducible to articulate or explicit form in its attempt to remove paradox from the equation. Such thinking is exemplified in the scientific management of Charles Taylor and the ideas of Herbert Simon, who in likening the workplace to an information processing machine, opined that tasks within the organization need to be reduced to simple manageable parts that individuals could handle without reference to others or the work of the organization as a whole. Diverging from this way of thinking, according to the authors, is the concept of dialectical thinking which embraces paradox and change through a process of thesis and anti-thesis leading to a new synthesis which forms the new thesis and begins the cycle anew.

Not all western thought has excluded this other aspect of enquiry. Researchers such as (Polanyi, 1962) who brought to light the role of tacit knowledge which informs the individual in their inquiries and actions even when we are not actually aware of their influences or able to articulate them. His statement "we know more than we can tell" (Polanyi, 1967) exhibits a keen awareness of the inexpressible knowledge that forms part and parcel of the context of any form of enquiry.

---

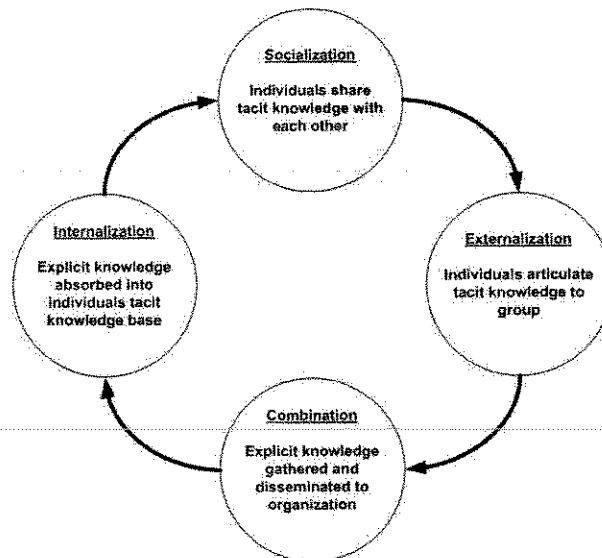
### Knowledge Spiral

The concepts of dialectical thinking and the explicit/tacit nature of knowledge when taken in concert result in an interesting formulation in terms of how knowledge is created and lays down a model of how it evolves within the individual and through the group and organization. Nonaka & Konno (1998)

describe a knowledge spiral, as depicted in Figure 2 which consists of four stages socialization, externalization, combination and internalization (SECI).

The initial stage, socialization occurs between individuals, as tacit knowledge is shared between them in a direct manner. Training periods, like those seen in crafts, trades and professions, where some form of apprenticeship is required to facilitate the transfer of tacit knowledge as know-how, from master to apprentice, along with the formation of an understanding of the context and mental models which form the basis of the area of expertise provide the most accessible examples of this process.

*Figure 2. Spiral of Knowledge Creation*



More generally, socialization can be seen in the development of relationships between co-workers as they come together as members of teams and projects, where they initially begin sharing experiences based on mutual interests and tasks. Another dimension is added where team members are drawn

from different disciplines. In these cases it becomes necessary for them to develop a common understanding of each others skills and perspectives arriving at a common basis upon which to interact. In these interactions tacit knowledge is shared between the individuals influencing the perspective of each of them with respect to their own knowledge, beliefs and understanding.

Externalization is the process of articulating tacit knowledge, that is held by an individual to a group of other individuals in a "self-transcending process" according to (Nonaka & Konno, 1998) and creating new knowledge in the form of concepts through the use of metaphor, analogy and models. Through the use of metaphor, (Nonaka & Takeuchi, 1995), the individual is able to articulate their tacit knowledge by elaborating a concept that captures and conveys the essential meaning of it by likening it to other things in the form of "figurative language and symbolism" (Nonaka, 1998). These conceptions are likely to contain contradictory meanings, which in fact act as a spur in furthering the conceptualization process and collective reflection upon it. Analogy on the other hand serves to identify the similarities and reconcile discrepancies within the elaborated metaphor. Modeling, takes the process a step further by translating the resultant concept into a logical framework.

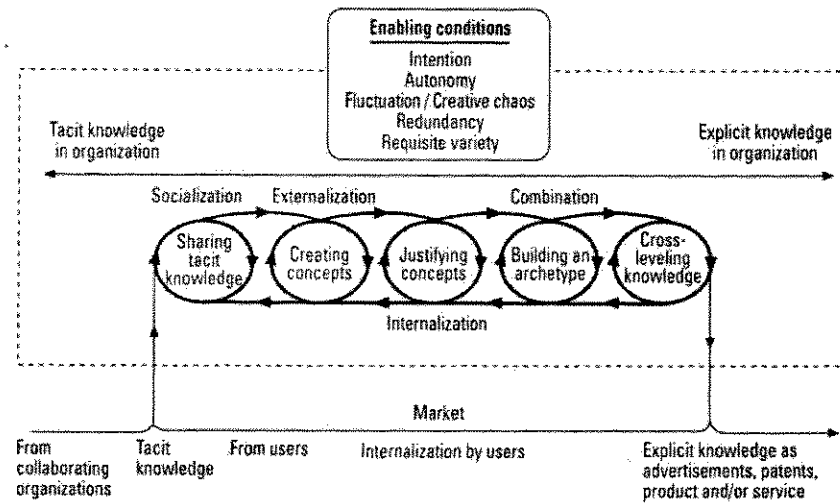
Combination represents a stage in the knowledge creation process where explicit knowledge is collected (from sources both internal and external to the organization), arranged and codified into a formal system and disseminated throughout the organization.

Internalization represents a reversal of the forgoing articulation of tacit knowledge, moving that articulated knowledge back down to the individual level. Through their exposure to and involvement with articulated knowledge the individual is able to make this knowledge their own. Knowledge within the organization evolves in this spiral manner being transformed from tacit to explicit and moving from the individual, through the group and into the very fabric of the organization where it spawns the next iteration.

#### Knowledge Creation Model

A more accessible articulation of how these ideas interact to create knowledge may be found in the “five-phase model of the organizational knowledge creation process” (Takeuchi & Nonaka, 2004; Ichijo, 2004) as it relates the somewhat unfamiliar concepts to a process that should be more familiar to the reader. Within this model as shown in Figure 3, five phases are identified in order to make the process more transparent to those who will be engaged in it.

Figure 3. Five-phase Model of the Organizational Knowledge-creation Process



From "Hitotsubashi on Knowledge Management; Hirotaka Takeuchi and Ikujiro Nonaka; Copyright © 2003." This material is reproduced with permission of John Wiley & Sons (Asia) Pte Ltd.

The phases can be briefly described as follows:

1. Sharing tacit knowledge, akin to the socializing process, starts when individuals within the organization spurred by some recognized internal or external need begin to discuss the problem from a variety of perspectives resulting in a more fully canvassed understanding of it and the potential for successful a response.

2. Creating concepts, representing the externalization phase, takes hold as the individuals collaborate to develop concepts which embody the tacit knowledge they have shared and evolved through the use of metaphor and analogy into some practical artifact of their conception.

3. The concept is then justified in such terms as are necessary for the organization to approve or deny continuation of the project.

4. The approved concept is then developed into a prototype stage which delivers a practical model of the proposed product.

5. Knowledge of the product is then propagated throughout the organization and potentially to interested external parties to make them aware of the new product and solicit their input.

This process is much closer to the software engineering paradigm where a solution would move from (1) identification of a need and initial investigation of solutions, (2) the development of a request for proposal or specification, (3) developing a financial case for the project and gaining management support, (4) building the solution, though theoretically a prototype system could be developed at this point to use in the solicitation of further input, and (5) deploying the system to the organization. Presumably the activity in phases 1, 2 and 4 would include input from the prospective customer population and the final phase would present them with a system shaped by their perspectives.

#### Knowledge Enablers and Barriers

Knowledge creation is a complex process which does not lend itself well to traditional methods of management. As has been shown in the previous paragraphs it is an organic process that grows out of the individual and their involvement with other individuals and groups and ultimately with their respective organizations. Rather than trying to manage the creative and innovative process, more success can be achieved through the fostering of an organizational environment or culture in which knowledge creation can take hold. An identified set of enablers and barriers within most organizations exist which can be

nurtured and overcome in order to create such a culture. The following enablers represent some of the ways in which the knowledge creation process can be fostered. In reviewing them one should keep in mind that they should be taken in concert rather than in isolation as they have are closely interrelated and produce a cumulative effect.

### Enablers

#### Fluctuation / Creative Chaos

Fluctuation within the organization, controlled and driven by management, allows the individual on their own and within a group to reflect upon and begin to question our perspectives and assumptions in relation to the normal way of doing things. While being initiated from within, the motivation for such introspection may come from within or outside the organization. Creative chaos on the other hand is used to spur the organization to engage in introspective activity in reaction to some crisis. In describing the alternative (Takeuchi & Nonaka, 2004) state:

This approach is in sharp contrast to the information-processing paradigm, in which a problem is simply given and a solution found through a process of combining relevant information based upon a preset algorithm. Such a process ignores the importance of defining the problem to be solved. To attain such definition, problems must be constructed from the knowledge available at a certain point of time and context.

By overcoming preset responses to the situations that are encountered the ability to achieve more creative and adaptive reactions will be achieved.

### Teams / Projects

Teams provide a central focus and context for individuals to come together and engage in knowledge creation for a specific purpose set out in their project mandate. The creation of cross-functional teams, according to (Newell & Huang, 2005), produce teams composed of individuals with different skills, backgrounds and knowledge. When composed carefully and allowed time to establish a base of common knowledge and understanding of each other these teams can produce very creative and innovative solutions arising out of the interplay of their diverse knowledge, as well as new knowledge that arises from their interaction. The efficiency of these teams may be impeded depending on the degree of separation between their fields and the difficulty in spanning such differences. In fact the success or failure of such teams often depends on their ability and willingness to learn from each other through, as (Cicmil, 2005) offers, the use of reflection to understand events and use this as a guide for their future actions.

### Creative Abrasion / Requisite Variety

Creative abrasion, according to (Leonard & Straus, 1998), involves the bringing together of individuals with different perspectives. These differences manifest themselves in the cognitive differences (e.g., experiential versus abstract thinkers) and ways in which people approach a problem. In bringing these diverse perspectives and personalities to the table we are also likely to encounter interpersonal conflicts requiring time to not only take care in the selection of team members, but in managing the interactions of its members.



### Reflection

Much knowledge creation occurs through reflection described by (Nonaka, 1998) as a place in time and space devoted to the understanding and evaluation of what actions have been undertaken in the past, their results, and their potential and consequences when used in the future. Without such examination, past and future actions are taken in a vacuum, with as it were a complete loss of knowledge. Perhaps a more important value of reflection is its ability to provide direction to the organization in times of chaos by permitting it to find new knowledge in the midst of change by challenging its orthodox views and responses. Reflection may also be a more effective way to manage projects where a high degree of uncertainty exists. According to (Gustafasson & Wikstrom, 2005), by allowing a more intuitive approach, as opposed to a rational one, we are better able to fine tune our responses to change through the unrestricted selection of informational sources upon which to base our decisions as to how we can reach our goal. The rational approach is seen here as being too encumbered with mechanistic rule following and isolation from reality to function effectively when change is the rule.

### Redundancy

Redundancy does not imply a surplus of resources, but rather the adoption of an approach where individuals are in constant contact across departmental or functional boundaries. This sort of redundancy allows for the constant transfer of tacit knowledge between the individuals through the creation

of common frames of reference. It may also include strategic rotation through departments and inter-team competitions on the same project (Nonaka, 1998).

#### Knowledge Integrators, Activists and Collaborators

Knowledge integrators play an important role in the knowledge creation process. These individuals cross organizational boundaries to gather knowledge from various areas and disciplines, combining them to achieve a new level of understanding or application within the organizational context. In his tenure as general manager of IDEA (Kelly, 2005) has identified a collaborator as a person who is able to transcend themselves to become a principle creator and motivator of teams. Collaborators believe that by working together towards some purpose will result in better outcome than could be had otherwise. Collaborators also have the effect of gelling team members into a cohesive unit which can mitigate the effect of individual resistance to change that would hinder the outcome of their activity, while at the same time coaxing the best out of them.

In *The Tipping Point*, (Gladwell, 2002), the author investigates various social phenomena through the lens of epidemiology. We are then introduced to three key players. Connectors are people who have an unusual capacity and interest in forming relationships with others and maintaining them, passing on bits of information they may think are of interest. Mavens are people who develop a deep understanding of a particular subject and also are driven to spread their knowledge to others. Salesmen are described as individuals who are particularly able to utilize both verbal and non-verbal cues to convince others of their points

of view. Each of these people exemplifies the knowledge activist in their ability to collect and disseminate knowledge.

### Promote Conversations

The promotion of conversations within the organization is crucial for the creation of knowledge. Conversation is the foundation of relationships that naturally occur amongst co-workers. Such relationships lead to further enhancement of the organization as knowledge is shared amongst individuals and groups. The promotion of conversation and thus relationships beyond the immediate group is also important for the purpose of fostering the breakdown of organizational barriers. The concept of water cooler conversations, as a medium where individuals can congregate and discuss their projects and exchange ideas can be taken even further with the idea of talk rooms (Davenport & Prusak, 1998) which are specially configured to enable and enhance the growth of conversations related to one's own project, the projects of others and the organization in general. Similarly the use of knowledge fairs to allow different groups within the organization to showcase the projects that they are involved in to other groups, which might never have been informed of them, is one way to promote inter-organizational awareness and cross leveling of knowledge (Davenport & Prusak, 1998). In promoting conversations we must also strive to create a welcoming environment for new and novel ideas. Kelly (2005) describes the "devils advocate" as one of the most destructive persona's that can inhabit the creative process. By looking for the flaws and potential downsides of every new idea they have the effect of chilling or killing the entire process. This is one

individual to be kept at bay. One effective method of quieting the inner devils is the adoption of a rule when engaging in idea generation that any objection to an idea must be accompanied with a reason for the objection.

### Organizational Barriers

The sense of organizational culture, (Ichijo, 2004) tells us, that each individual possesses is acquired through their involvement in the language, stories, perceptions, proceedings and paradigms of the organization. As with any complex social structure the individual takes their cues as to what constitutes proper behavior in the context of the organization. Such cues filter down from the top of the organization from the leaders of the enterprise and more immediately the managers of its various components. Where not explicitly laid out in procedure manuals or training sessions, messages are sent through the medium of the language used to describe the organization its goals or mission statement and the everyday tenor of conversation. Organizational stories also inform the individual as they relate the history of the organization and those who have been revered or shunned for their actions. More subtle messages are received, whether they are intended to be or not, through the perception or sense of how the organization wants its members to function. All of these aspects of an organization form a reflection of its overall paradigm and to the extent that they do not evince a desire to foster an open and sharing knowledge culture they will create and reinforce a barrier against it.

## Individual Barriers

### Threat to Self Image / Limited Accommodation

Individuals and in particular professionals define themselves, rightly or not, by their skills, knowledge and reputation within their chosen profession. Knowledge sharing in this context represents one threat to their self image by not only by asking them to reduce their knowledge to expressible form but in that their established beliefs may come into question as their knowledge enters the public forum. Without the ability to accommodate and change in response to these new ideas (Ichijo, 2004) states that many individuals will erect barriers to fend off the perceived attack on their knowledge and limit their involvement in knowledge sharing activities. A further response can be seen in the use of defensive reasoning (Argyris, 1998) as a response that is likely when one's beliefs come into question. In this scenario the individual or group will erect arguments which deflect and avoid the admission that their own actions are the cause of a problem or that changing them would result in a more beneficial outcome.

### Knowledge Hoarding / Knowledge Walkout

Knowledge hoarding (Tiwana, 2002) is an unfortunate if natural result of the real or perceived threat to an individual's self image. Where an individual's value to an organization is measured by the knowledge they possess, as related to them in terms of evaluation and promotion, the benefit to them of sharing their knowledge will be far from apparent. In fact they may see it as giving away what value they do represent to the organization. As a result they will resist attempts to extract their knowledge impeding knowledge creation. Another serious

consequence of knowledge hoarding is the potential for a “knowledge walk out”. When knowledge of a particular system or process is concentrated in the mind of a single individual that individual’s departure from the organization will pose a significant problem. These difficulties arise when the organization has not taken the appropriate steps to ensure that the departing individuals’ knowledge has been transferred, to some degree, either to another or sufficiently articulated and captured to enable its transmission.

#### Not Invented Here

The idea that knowledge created outside of the immediate organization, even to the level of inter-departmental barriers, is of no utility can prevent the application of previous knowledge creation. This sense of “not invented here” should be well understood by information technology professionals who exemplify this sort of behavior when they invest valuable time in creating their own versions of algorithms, classes and even programs that could be more readily obtained and adapted through code reuse simply because they don’t trust code repositories or even the programmer across the hall.

#### Not Enough Time

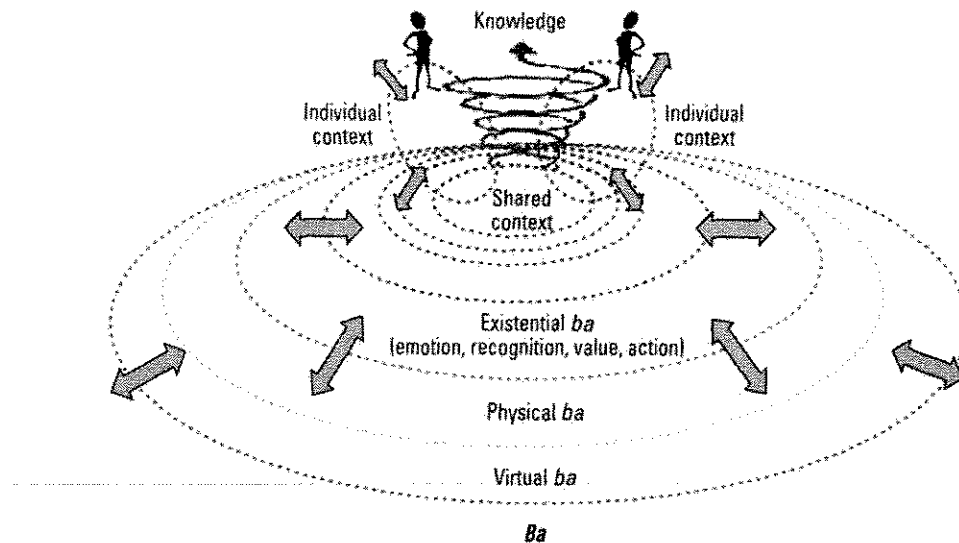
Where sufficient time is not allocated within the processes of the organization for the promotion and practice of knowledge creation the demands of routine or normal functions will preclude, to a large degree, its adoption. Enabling the basic requirements for knowledge creation in terms of the creation of relationships between individuals at both the local and inter-departmental level as well as many of the enablers, such as reflection, conversations, and

knowledge fairs require a commitment of time from both the individual and the organization, over and above the daily routine.

## BA

One of the chief components of knowledge creation is the Ba. Ba, as (Nonaka & Toyama, 2004) explain, meaning a “place” in both the literal and figurative senses culminating in the creation of a space in which an individual’s context can interact with the contexts of others resulting in “a phenomenological time and space where knowledge as a stream of meaning emerges”. The emergence of the Ba within individuals, working groups, project teams and other organizational units begins to occur as a self organizing phenomena (Nonaka & Konno, 1998) which allows for the transcendence of one’s own perspective through exposure to and reflection upon the experiences, values, meanings and mental models of others. In doing so the Ba, as shown in Figure 4, has the effect of energizing and advancing both individual and collective knowledge. The leaders’ role within the Ba, according to (Nonaka & Toyama, 2004), is to enable and foster the formation of these groups by providing them with relevant enabling forces like autonomy, creative chaos, redundancy, requisite variety as were canvassed above and the exchange of explicit and tacit knowledge that emerges from them.

Figure 4. Conceptual Representation of Ba



From "Hitotsubashi on Knowledge Management; Hirotaka Takeuchi and Ikujiro Nonaka; Copyright © 2003." This material is reproduced with permission of John Wiley & Sons (Asia) Pte Ltd.

The Ba exhibits 4 characteristics, as described by (Nonaka & Konno, 1998), each of these characteristics parallels the knowledge spiral which was explained above and the import of the enabling factors which have just been reviewed. Originating Ba occurs where an individual shares their meaning, feelings, emotions and mental models with others and begin to establish a relationship of mutual trust and respect. The interacting Ba is a more artificial creation being a group of people, specifically selected for their knowledge or skills, brought together to work on a particular issue or project. Critical at this stage it that "[t]hrough dialogue, individual's mental models and skills are converted into common terms and concepts. Two processes operate in concert: individuals share the mental model of others, but also reflect and analyze their



own” (Nonaka & Konno, 1998). Through conversation and the use of “metaphor” the interacting Ba becomes the place within which an exchange of tacit knowledge and knowledge creation can begin to occur. The cyber Ba represents the transformation of the tacit knowledge made explicit through the interacting Ba to an explicit form which can then be disseminated beyond the immediate group. The exercising Ba provides for the conversion of new explicit knowledge into tacit knowledge in the individual.

### Software Engineering and Knowledge Management

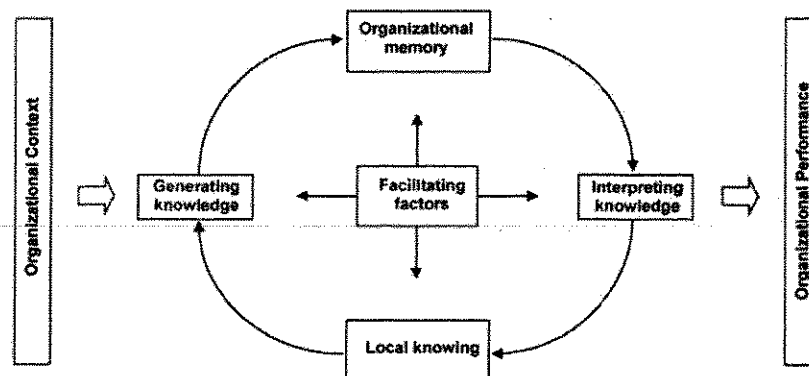
Software engineering in both its practice and purpose deals with a wide range of data and information which must be transferred and transformed into knowledge for this to be accomplished. Software engineering is, as (Edwards, 2003) tells us, is a knowledge intensive activity requiring that an approach to the solution of knowledge management issues within it proceed from both the perspective of codification to ensure the capture and retention of project knowledge and personalization focusing on the individual and their role in the process. One difficulty that all project based endeavors present is the lack of an ongoing organizational memory, as by their very nature projects are time limited and generally focused at accomplishing a singular task as (Love, Fong, & Irani, 2005; Bresnen, Edelman, Newell, Scarbrough, & Swan, 2005) point out, and therefore require directed efforts to capture and retain the knowledge they create.

Lindvall & Rus (2003) make a case for knowledge management as a means to capture two distinct sets of domain knowledge that are critical for the successful execution of software development initiatives. The first domain is that

of technology and the projects ability to not only implement solutions with the best and sometimes latest technology available, but also to plan the implementation. The second domain is the problem domain, and its policies and customs, which needs to be understood to formulate a correct solution as well. The authors point out that knowledge within software projects is to a degree unique as much of it is already captured in various work products like, project plans, requirements, design, and other documents which can form the basis of a knowledge management initiative to disseminate this knowledge to other software engineers. The ability to locate tacit knowledge in the mind of the knower, they argue, must also be enabled through competency management and the development of not only a means to identify and access them, but also to know when a deficiency in knowledge exists. The need to capture the reasons for the selection of a particular design, among many considered, is one area that is not well covered leading to a loss of "product memory". At the same time it is necessary to be able to trace the systems requirements and defects into the coded system. The authors stress the need to avoid the mistake of focusing on technological solutions advocating the careful cultivation of a culture of sharing through the use of lightweight processes which allow knowledge capture to evolve naturally. Challenges to the adoption of knowledge management are similar to those enumerated in the previous chapter and include the concerns of not enough time, not invented here and knowledge hoarding, which can only be overcome through a cultural shift.

A model advanced by (Dybå, 2003), shown in Figure 5, applies the concepts of knowledge creation to the software engineering field, stressing the need to integrate knowledge within the software development process. This idea is echoed in (Rao, 2003) who believes that many of the knowledge management behaviors can be directly embedded into the development process itself. The facilitating factors, Dybå describes, are very similar to those espoused by previous authors with respect to knowledge creation and they will not be repeated.

*Figure 5. A Dynamic Model of Software Engineering Knowledge Creation*



From "Managing Software Engineering Knowledge", Copyright © 2003, p. 97,  
Tore Dybå, "A Dynamic Model of Software Engineering Knowledge Creation"  
Figure 5.1. With kind permission of Springer Science and Business Media.

Local knowing describes the interactive nature of developers as they collaboratively work towards an understanding of the problem domain and arriving at practical solutions within it. In doing so the software organization will also be engaged in the creation of new knowledge of how they do so, adding to

their knowledge about software engineering. The author sees this as an improvisational activity drawing on previous experience and current context. Generating knowledge is carried out in much the same way as was described by (Nonaka & Toyama, 2004) through dialog, collective reflection and the use of divergent and convergent thinking methodologies (roughly equivalent to the dialectical thinking through metaphor and analogy) resulting in concepts and models that can be packaged to share with others. Organizational memory serves as a means to justify and transmit the new knowledge and past knowledge into the awareness and context of those who will have need of it. Justification must also serve as a basis on which the knowledge can be re-contextualized into new situations. Interpreting knowledge completes the knowledge creation circle and represents the point at which organizational memory is interpreted back into the local knowing context. Within this context it is both the individual and the larger team that must engage in collective interpretation to make sense of the knowledge in the context that they find themselves and the problem they are trying to solve. Dybå sees this as knowledge that is jointly constructed rather than merely an application of previously captured knowledge. In this sense he equates it with a re-experiencing of the knowledge discovery or exploratory experience.

Dutoit & Paech (2003) identify two domains in the software engineering context, which they term, the application domain which is the preserve of the customer who understands why they are asking for a particular feature or function and the solution domain in which the software engineers have specific

knowledge of the structure of the system. They maintain that these two domains are exclusive and that a complete solution can be achieved only through requirements engineering over the life-cycle of the product and the close collaboration of diverse individuals. In order for this collaboration to be successful in the long run they describe 5 essential knowledge needs: Sensitivity Characterization: which evaluates the potential for future change in a given requirement; Rationale: the reasons for decisions made, which can help when changes are considered later by eliminating the need to revisit options that had previously been canvassed; Pre-traceability: the ability to locate the person who requested a feature in the first place; Post-traceability: provides the ability to trace a requirement throughout the software development life-cycle and Change Impacts: the scope of impact that a change to a particular part of the system could have to other components of it. While having each of these knowledge sources available would give us a much fuller picture of how the system was conceived and created, it would also seem to create a huge overhead in terms of the knowledge capture effort required to achieve it.

#### Software Engineering and Knowledge Management Technology

---

Many efforts have directed their attention to the creation of systems to manage knowledge meeting with varying degrees of success. Davenport & Prusak (1998) allude to the subtlety of expert knowledge being too complex a thing to be captured in a system, though they are referring to earlier attempts to create "expert systems" their words are no less applicable to many of the efforts

since. In fact they go on to say that technology can neither make an expert, a learning organization or a knowledge creating company, nor make people use it.

A study by (Dingsøyr & Conradi, 2003) presents us with a knowledge management process that tries to combine the codified and personalized view of knowledge just described. To accomplish this a system, Knowledge Flow, was created and made accessible over the internet comprised of a knowledge repository (Well of Experience), used to capture documented knowledge, communities of knowledge workers, workspaces based on discussion boards and work assignment tools, and a skills manager which linked workers to their various areas of expertise and inventories of their skills. After conducting interviews and reviewing the usage logs of the system they found that the knowledge repository was used primarily by the developers to answer to deal with some specific technical problem or get an overview of an area they were unfamiliar with. Experts used it to try and avoid having to answer the same queries multiple times, but at the same time many would simply locate the expert and go to them directly rather than access the information contained in the repository. Difficulties in searching the repository and a lack of captured knowledge were also documented. The Skills Manager was also use to locate experts, but was primarily used for resource allocation.

In (Kochikar & Suresh, 2003) the application of knowledge management at Infosys is detailed. The projects purpose was to enable the organization to raise their level of customer service by achieving greater facility in terms of learning and decision making under the pressure of the rapid changes in technology.

Careful consideration was given to the role of people in that they would both support the system by contributing to it and using it. The concern for contributions centered on the ideas canvassed previously such as knowledge hoarding, while usage concerns were addressed by the architecture which will be addressed shortly. As an added incentive to use the system, contributors are awarded points (KCU's) for their submission of articles and can gather further points from users of their articles if they see fit to award them. The points can then be turned into a certificate and redeemed for goods through a rewards scheme. The use of subject matter experts to review submissions also adds credibility to the articles. In order to facilitate use of the system its content architecture was broken down into a hierarchical taxonomy using a variety of keywords to allow the user to find the topic they are looking for. Within each topic several content types are available such as, case studies, frequently asked questions, bodies of knowledge and the like. Additional services such as a people locator and people-knowledge map add further resources and relevance to the system. The system is delivered through a portal, given the global nature of Infosys's business and allows the user to customize it to their particular set of knowledge requirements. In assessing the success of the project the authors found that 99% of respondents thought that knowledge management was essential, with 71% believing that Infosys encouraged participation in it and that 71% believed it had saved them a day of research, while 14% believed they had saved at least 8 days.

## Conclusion

Throughout the knowledge creation process we can see the flow of tacit and explicit knowledge as it moves through the contexts of the individual, group and organization. The idea of “conversation” is a key facilitator of this movement, laying a foundation of understanding and trust which are fundamental to the relationships between individuals that are required for knowledge transfer to begin. Through the use of metaphor and analogy individuals with widely divergent knowledge, experience and skill are able to bring their unique perspectives to bear on the problem that they are attempting to solve by evolving a transcendent dialog centered on a common set of concepts and models. Through these concepts and models they are able to understand each others perspectives and exchange, at least to the degree necessary, elaborations of their tacit knowledge allowing them to affect each others perspectives through a process of reflection. It is in this exchange that new knowledge emerges. As each individual reflects on their own knowledge in light of that of others, new combinations and permutations are uncovered.

The degree to which a process within a group or organization promotes a culture of knowledge creation by supporting its enabling factors and overcoming the barriers to it, is the measure by which it can be evaluated as to the efficacy of its knowledge management practice. This can also be seen in the ability of an individual, group or organization to adapt to and accommodate change which requires a revaluation of their knowledge, as “fundamental true belief”, and allow them to not only deal with complexity and chaos, but thrive on it.



## CHAPTER IV

### AGILE METHODS AND KNOWLEDGE MANAGEMENT

#### Introduction

The author contends that traditional methodologies of software engineering suffer from a knowledge deficit due to the manner in which they approach software development. In its attempts to perform upfront requirements analysis, design and planning it ignores the reality of change, feedback and learning about both the product it seeks to deliver and its own processes. This is not to say that it can't take these issues into account, but that it does so only at great and sometimes insurmountable costs. The Agile methods have a tendency to ameliorate this knowledge deficit through their practices which are more closely aligned with theories of knowledge creation espoused by the work of Nonaka and Takeuchi. Rather than proceeding through a review of several of the Agile methodologies this chapter, the knowledge creation concepts and models that were elaborated will be applied to them. By looking at the Agile methods in this manner the author seeks to evaluate whether they already embody the use of knowledge management through their processes, how project knowledge is collected, disseminated and reflected upon and how this project knowledge influences their ability to deal with complexity.

#### The Agile Movement

While many of the methodologies that would form the foundation of the Agile Movement were all ready developing, Crystal Clear (Crystal) (Cockburn, 2005), Extreme Programming (XP) (Beck, 2002; Beck & Andres, 2004; Jeffries,

Anderson, & Hendrickson, 2000; Marchesi, Succi, Wells, & Williams, 2002; Wake, 2001), Scrum (Schwaber & Beedle, 2001) among others, the creation of the Agile Manifesto marked the coming together of the founding members of the Agile Alliance and the following expression of their common views.

#### Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools  
**Working software** over comprehensive documentation  
**Customer collaboration** over contract negotiation  
**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, the above authors this declaration may be freely copied in any form, but only in its entirety through this notice. (Beck et al., 2001a)

Each of the four statements in the manifesto describes a high level view of the principles (which will be elaborated in the rest of this chapter, and are attached as Appendix A) that form a common thread running through the methodologies. The principles can be seen as a reaction to the increasing degree of formalization and systematization of software engineering taking place at the end of the 1990's. The signatories in light of their experience and practice of software development advance a set of practices based on the importance of

communication between people, the essential products of the software development process and the inevitability of change during that process.

### Knowledge Creation in the Agile Methods

Knowledge creation occurs at many levels during software development process, and the author would argue that the act of creating software is itself a knowledge creation activity. The creation of knowledge in the Agile methods should follow much the same path as the one described by in the previous chapter as it moves through the four stages of socialization, externalization, combination and internalization.

#### Socialization

Socialization seeks to provide the individual the opportunity to build relationships with others. Within the Agile methods this is supported by many of the practices that are followed. The establishment of a foundation of trust is essential in beginning the socialization process. The XP concept of “whole team” (Beck & Andres, 2004) provides focus on the purpose and needs of the team rather than the individual while Crystal’s “personal safety” (Cockburn, 2005) allows the individual to express their true feelings about the process and their role in it without fear of reprisal.

A common feature of the Agile methods is the idea that the team should be co-located. This not only aids in the development of a teams identity but facilitates communication and collaboration between team members. Co-location also leads to the notion of osmotic communication (Cockburn, 2005) which asserts that the flow of communication between individuals within hearing of the

group will not only be absorbed by the parties to the conversation but also by the other members of the team adding to their information and knowledge of the projects design, direction and difficulties.

Pair programming, while being one of the most contentious XP practices, is also one of its most beneficial in terms of socialization. In this sense we are not looking at the actual code produced, which the author considers part of the combination phase, but as a practice which fosters and creates the relationship of the pair. Pairing has also been found to decrease the time it takes, and hence the cost, of adding new members to the team at its later stages (Williams, Shukla, & Antón, 2004). This activity allows for the direct transfer of knowledge both tacit and explicit at both the professional and project level as well. Canfora, Cimitile, & Visaggio (2004) have also found that “pair design” can have a similar impact on individuals working in pairs in terms of building their design knowledge.

#### Externalization

Externalization being the process of articulating tacit knowledge to explicit knowledge and involves itself heavily with the elicitation of the customers' knowledge of their requirements. The author also includes under this category the development of feature and release planning as it should be seen as part and parcel of the expression of the customers' rational and need for the product.

Crystal uses a process called Chartering, as described by (Cockburn, 2005), as its initiation process to bring together the team, which at this point is limited to the project sponsor, lead designer and perhaps an expert user to complete what it terms an Exploratory 360° which includes:

- Adapting the methodology (methodology shaping) to fit the projects circumstances and creating a high level project plan
- Determining what the project offers in terms of business value
- Defining a preliminary set of requirements that will be needed to provide the desired functionality
- Conducting a survey the domain model and the prospective technologies to be used is conducted to ensure that they can in fact succeed
- The creation of an initial project plan is then carried out from the determined requirements or stories and planned using the blitz-planning technique (described below)
- At this point the project sponsor will be able to decide if the project should go any further

If the decision is made to proceed the team will then start the planning phase. Blitz Planning (Cockburn, 2005) incorporates the priorities of the customer, the estimates of the developers and a joint responsibility to arrange the work in the most efficient and valuable configuration possible. It begins by having the customer and team create index cards with the tasks that need to be completed. The task cards are then laid out in dependency order, tasks which can proceed in parallel are laid out beside each other, while duplicates are removed. Tasks are then reviewed to make sure that no tasks have been missed and new tasks may be added in response to the ones already created. Task

estimates and assignments are then made and further refinements to the dependencies. Of particular interest are the ideas of:

1. The walking skeleton which represents a prototypical system capable of performing some rudimentary processing from end to end of the intended system, a proof of concept.
2. The earliest useable release is the barest set of features which could be of use to a customer and also marks the point at which the wider customer community will be able to see the product and comment on it.
3. The earliest revenue producing release which is when sufficient functionality has been implemented to justify a wider rollout.

These cards should be prioritized so as to facilitate reaching these important objectives. Other potential releases can then be identified by looking for logical groupings of functions and features. Finally the plan is captured.

XP begins with the customer and prospective system users, assisted by the development teams' project manager and interaction designer arriving at a suitable metaphor which will serve as the overall vision for the system. Once this has been accomplished they can begin to develop stories, narrative descriptions articulated by the customer, with the assistance of the team, of the essential functions that the system will carry out. Developers are then asked to work up estimates based on time to implement for each of the stories, breaking them into tasks where they exceed the length of a single iteration. This has the effect of giving the team a sense of what the customer's expectations are, but also allows

the team to inform the customer of the time, cost and technical difficulty of the prospective stories.

Planning then proceeds for the release (which sets of features will be developed before releasing the system) based on quarterly and iteration time spans. The customer selects from the stories those which they would like to have implemented in the next iteration and over the longer term, applying primarily a metric of what represents the best value to their organization to prioritize the implementation of functionality. Where a set of stories amounts to more work than the team can reasonably handle in the current iteration the customer and the team must negotiate to reduce the number of stories by re-prioritizing or shifting them to later iterations.

Key to this stage of the knowledge creation process is the interaction of the customer and the team. The articulation of requirements/stories and planning requires the collaboration of the customer and the team in bridging their individual contexts to create a common context where each understands their needs, capabilities and limitations. In doing so the customer informs the team not only of the products technical aspects but also which aspects are of greatest value to the organization. The team in turn is able to produce a more realistic plan for the implementation of the project, having been informed of both the content and context of it.

#### Combination

Combination involves the embodiment of the tacit knowledge made explicit in the externalization stage and the individual knowledge of the team and

its members in some form of intelligible form. In the software engineering context this can take no higher form than the delivery of functioning software that meets the customers' requirements. To achieve this XP follows an incremental and iterative approach. Crystal follows a similar trajectory differing for the most part only in terminology (it may in fact adopt XP as its implementation strategy) and the length of time an iteration lasts. Differences will therefore only be noted where they introduces a significantly different method or a more informative explanation. The discussion that follows can be seen as the activity that would take place during a single iteration culminating in a delivery or release of the software product.

The use of informative workspaces, serves to remind the team of their goal and the progress they are making (or not making) towards it. Crystal calls these "information radiators" (Cockburn, 2005) which are visible to people both inside and outside of the development team, but really are of more use to the latter, as the team should be well aware of where they stand.

XP proceeds from a test driven development philosophy. As programmers begin the iteration they will design tests based on the functionality that the stories describe. This has a three fold benefit. Firstly, it ensures that the developers have thought through what the story actually requires and will design their tests to meet the minimally necessary functionality of the story. Secondly, having created the tests they will be able to easily determine when they have met those requirements. Finally, they are contributing to a body of tests that will assist them



when they integrate their code into the code base as well as providing a baseline against the effects of future refactoring can be verified against.

Design in XP is carried out in incremental stages in order to avoid creating designs for features that may or may not be implemented in the future (Larman, 2003), with just enough design undertaken to implement the immediate story or task into the system as a whole. The efficacy of pairs conducting design work lead to mixed results in a study conducted by (Al-Kilidar, Parkin, Aurum, & Jeffery, 2005) where final year students were given six weeks of training and then asked to design a web based project management tool. While the design of simple modules was better, more complex ones were not.

Pair programming (Beck & Andres, 2004) is conducted by two programmers, and includes the foregoing test and design development, working at a single computer screen one programmer enters the code (the driver), while a second programmer (the partner) watches to ensure that they are adhering to the story, coding guidelines and the tests they have developed. At the same time the partner may offer suggestions as to how the code might be implemented offering feedback or the opportunity to discuss what the driver is trying to do. Rotation between the driver and partner is also encouraged. Crystal makes use of side by side programming (Cockburn, 2005) which provides some of the benefits of pairing (code review, testing, discussion and coaching) while the programmers can work on separate tasks at the same time

XP aims at continuous integration. By combining the code that various sets of pairs are creating, the system codebase is kept up to date and problems

between conflicting modules are identified and can be corrected early (Beck & Andres, 2004). To assist the team in this stage of development some tools are considered essential. Crystal and XP recommend the use of automated build and testing, while Crystal adds configuration management as well.

The point at which deployment or delivery of the system occurs would generally be determined by the release plan that has been agreed to prior to the commencement of the iteration. It is more likely that a series of iterations would take place, depending on their length and frequency, before any deployment to the customer base. (Beck & Andres, 2004) offer that an incremental deployment offers a means of avoiding the risk of deploying the entirety of the system all at once. Through this strategy of incremental deployments not only can the customer begin to obtain value from the system, but the team can gather feedback from the user population as to how the system serves their needs. Cockburn (2005) argues that frequent delivery (less than 4 months in between) to a real user is important for a number of reasons: it provides the customer with a measure of progress, developers have a target to focus on and users have the opportunity to review what has been produced at their behest and comment upon it. It is this last part that is most important for the combination stage, the ability of the team to disseminate the results of their codification of explicit and tacit knowledge collected during the iteration and made manifest in the system delivered.

### Internalization

Internalization represents the reversal of the externalization stage. Within the software development process this takes place after the product has been delivered and before the next iteration.

In XP post Iteration reflection (Beck & Andres, 2004) takes place at the end of the iteration and is a good time to reflect on not only the work that has been accomplished, but more importantly how it was accomplished. Were there things that could have gone smoother? Does our development model need any tuning to make it better? Such questions can provide a basis for the team to reflect on what they have just done and make any needed adjustments for the next iteration. It also considers feedback about the product and what knowledge can be gained from any failures it may have encountered along the way.

Crystal makes use of two specific devices to incorporate internalization into its methodology. The first is the keep/try reflection workshop (Cockburn, 2005), which can be used during or after an iteration, is used to reflect on the way the team is functioning and whether there are any persistent problems that need to be addressed and whether there are practices that should be kept or new ones that should be tried. The practices referred to relate to the conventions of the Crystal methodology or others that the team believes would improve their environment. The second device is that of reflection on the delivery. In this activity the team not only reflects on how the delivery proceeded, but actively seeks input from the users of the system to ensure that they are delivering what the users needed and expected.

## Agile Methods: Enablers, Barriers and Ba

The Agile methods provide a host of guidance with respect to enabling software development, removing barriers to it and creating a context where this activity takes place. In the previous chapter a set of enablers, barriers and the concept of Ba were introduced at this point they will be revisited in the context of the Agile methods.

### Enablers

#### Fluctuation / Creative Chaos

Fluctuation is described as being a lever activated by management to cause a revaluation of the normal routine of the organization motivated from within or outside of it while creative chaos is motivated by a far more serious crisis due to the latter. During the lifespan of an Agile project changes can occur where the planned path of development is forced to deviate due to a change in the features required for the product as a result of a change in the customers' business environment. The ability to add, remove and change the order of stories, the use of short iterations and incremental design allows the customer to make needed changes without resulting in major disruptions. In fact both XP and Crystal make the ability to accommodate change part of their core principles.

#### Teams / Projects

While this category is implicit in the Agile methods it should be remembered that the team is not composed solely of programmers. The customer (sponsor, expert or both) plays an important role in the process as it is they who define the product's specifications and the importance of them to the

rest of the team. It is they who tell the story, the rest merely act it out. The importance and therefore difficulty of this role cannot be underestimated. Looking at the customer role in three XP developments (A. Martin, Biddle, & Noble, 2004) found that the customer role by its nature results in a high degree of stress, as the customer is responsible not only to the team (stories, financing and reporting) but the business as well which may make the role unsustainable over the long term. Through its attention to socialization and externalization the Agile methods ensure that the team will develop into a well functioning group, learning from each other and engaging in creative development. According to (Highsmith & Cockburn, 2001) Agile software development practices are generative rules rather than a strict prescription as to how to do software development. In this sense they provide an adaptive environment in which self-organizing teams can create their own process. As (Boehm & Turner, 2003) remark, plan-driven methodologies tend to be all encompassing, intended to be tailored down to suit the situation, while Agile methodologies move in the opposite direction offering a minimalist approach and growing only when justification to do so can be found.

#### Requisite Variety (Creative Abrasion)

The composition of teams in the Agile methods consists largely of technical people. Within Crystal (Cockburn, 2005) identifies an executive sponsor, expert user, lead designer, designer-programmer and four subsidiary roles, a coordinator (filling the role of project manager), business expert (for questions about the business outside the users expertise), tester and writer. In

XP (Beck & Andres, 2004) identifies ten personages as the “whole team” (human resources has been omitted). Here we find:

1. testers: who assist customers define acceptable functioning and developers design the tests to achieve it, as well as system tests
2. interaction designers: assist in the development of stories and metaphors that embody the system
3. architects: shape the overall design of the system, wrestle it back into shape through large scale refactoring and find its weak points through stress testing
4. project managers: act as facilitators, passing information to the team and back out to the customer in the appropriate manner. They also engage in continuous planning as the project evolves
5. product managers: manages the writing of and acting upon stories in a sequence that makes both business sense and the teams progress manageable
6. executives: provide leadership and context to the team
7. technical writers: provide users with an idea of what is being developed and relate their concerns back to the team. They also produce user documentation and conduct training
8. users: are part of the story development and selection process
9. programmers: estimate stories, break them down, design tests, code and improve the system

This would seem to provide a reasonable degree of variety, though many of the participants are technical, they are sure to come from and hold different perspectives as to how to accomplish a given task.

### Reflection

As was examined above both XP and Crystal incorporate reflective practices into their lifecycles in order to not only improve their internal functioning but also to incorporate what they learn from the delivery of their product to improve both its delivery and its content.

### Redundancy

The mandatory inclusion of a customer representative under XP's methodology and access to an expert user in Crystal allows for the cross-functional nature of redundancy envisioned by Nonaka by achieving a team based level of boundary crossing.

- Knowledge Integrators, Activists and Collaborators

Redundancy and many of the roles just described which form part of the Agile methods incorporate to a large degree what is meant by the role of knowledge integrator / activist. Customer collaboration as designated in the Agile Manifesto plays a crucial role in determining the shape and outcome of the project.

### Promote Conversations

Conversations are one of the critical elements of the Agile methods as can be seen in many of the activities that were described above, including, sitting together, pair programming, the standup meeting, story development, and

reflection. This informal sharing of knowledge while avoiding knowledge loss according to (Chau & Maurer, 2004) creates problems for those who are not a party to the conversation but many still have an interest in the content of it (see the discussion of documentation below).

### Barriers

#### Organizational

The structure of organizational barriers, defined in the previous chapter, must be recast depending on the situation that an Agile project finds itself. The wholesale adoption of an Agile methodology within an organization has met with many forms of resistance from process issues, business perspectives and people (Boehm & Turner, 2005). Hodgetts (2004) describes adoption of Agile methods in some organizations as a disruptive change, requiring a more focused and slow introduction to be successful. Even where the project is a separate initiative there may still be a need for it to overcome difficulties in developing the requisite relationships with other entities in the organization be they technical or not if the organization has not adopted a knowledge management philosophy.

At another level, that of the team itself, resistance should not be a problem as the paradigm of the Agile methods is focused on language, stories and procedures that are in line with a culture of knowledge creation.

#### Not Invented Here

As was identified as a particular disease among programmers this syndrome may or may not be overcome by the Agile methods. One would hope



that the use of pair programming would make programmers more open to new ideas, other than their own.

#### Knowledge Walkout / Knowledge Hoarding

Each of these barriers is a concern where a single individual has in some way gained control over a particular area of knowledge important to the functioning of the organization. XP's use of pair programming ensures that at least two people have knowledge of any particular implementation of a single story; in this situation even if one programmer decides to leave there is at least one more that understands it. The same reasoning holds true for the concern that knowledge of the system will be hoarded. In fact it would be difficult for a single developer to hoard information about any part of the system for two reasons; first is XP's use of shared code and a single codebase (Beck & Andres, 2004), while the former is used to allow any pair to refactor the system when they identify a part of it that needs improvement, the latter is used to avoid multiple code streams and their attendant maintenance requirements. As an added benefit this makes the system transparent to the whole team. Similar problems may arise with the customer representative(s) on the team, but are not addressable here.

---

#### Not Enough Time

The processes, enablers and barriers to knowledge creation up to this point represent aspects of the Agile methods which are incorporated into their practice thereby alleviating the concern of having enough time.

## Documentation

Another aspect of the Agile methods is that of documentation. Due to the nature of the discussion surrounding this topic and the difficulty in classifying it as either an enabler or barrier the author has decided to treat it separately.

Cast as a barrier we find the arguments that were canvassed in the last chapter surrounding the need to capture the knowledge a project creates due to their temporal nature and attendant memory loss (Love, Fong, & Irani, 2005). The ideas of (Lindvall & Rus, 2003) with respect to the fact that project knowledge are at least captured in the work products of the software engineering process but are turned upside down in their veiled reference to the Agile methodologies which do not ascribe to such a full set of outputs. Raskin (2005) refutes the XP notion that documentation is an impediment to the real work of programming, stating that the “why” of a particular implementation, its rationale, is lost if a programmers code is not commentated.

As an enabler, (Beck & Andres, 2004) addressing the issue of permanent artifacts, insists that only code and tests should be maintained and that the organizational memory should maintain the projects history. Elsewhere he explains that the preference for direct communication was not intended to be an excuse for a lack of documentation, but goes on to argue that anything that does not add value to the functionality of the system is waste. Taken to the extreme some XP practitioners even eschew the use of code comments as they too require updating as the code changes, preferring to rely on “self-documenting code” (Schach, 2002). Ambler (2005) argues that documentation should not be

seen as a primary vehicle for communication, rather it should be lean, light and just good enough to facilitate the ongoing development effort without allowing the cost (time) to produce it outweigh its benefits. A major concern is the fact that if documentation is not constantly updated once produced, the information they convey will lose its currency almost immediately. Ambler goes on to list some forms of documentation that are worth while. Among them are design decisions which explain why a particular architectural or design was implemented and what other approaches were considered to facilitate a better understanding of the system and refactoring which could go back over discarded decisions if they are not made explicit. Melnik & Maurer (2004) explain that non-Agile software engineering practices treat knowledge as an object capable of being documented even though some 70% of such documentation has been found to be seriously compromised. They go on to explain that "highly abstract knowledge" needs to be transmitted over richer communication channels than documents offer in order to be effective. A study conducted by (de Souza, Anquetil, & de Olivera, 2005) provides us with insights into the use of documentation in the maintenance phase of the systems lifecycle. In their study of 76 software maintenance professionals (consisting of managers, analysts, programmers and consultants) they conducted a survey in order to ascertain their usage patterns of various software development artifacts. Their findings for object-oriented artifacts ranked as very important, source code and comments at 94.3% and 75.9% respectively, with the logical data model, class diagram, physical data model, use case diagrams and use case specification rounding out those artifacts that garnered more than a

50% mark. Crystal's (Cockburn, 2005) answer to documentation is that the team and sponsor decide, but then goes on to restate some of the arguments made by others that time spent on documentation will not be spent advancing the system, that documentation is never really up to date. Cockburn then allows that some record of the teams endeavor should be created, integrating the requested documentation as a distinct part of the planning process to account for the effort required to produce it. The actual artifacts, he argues, need not follow traditional methods but may include photographs of whiteboards, scans of hand drawn diagrams or even videotapes of people discussing the design of important parts of the system at the whiteboard (Cockburn, 2004).

It is difficult to determine whether the Agile methods approach to documentation is inadequate or not. XP is correct that the ultimate product of software development is working software, but without the ability to review the manner of its construction, the decisions taken and plans made, whatever learning that may have taken place along the way will live on only in the minds of those who originated it. In that case we are left to hope that a few of them decide to write it down and pass it along.

---

## Ba

The Agile methods outlined in the previous two sections fit well within the description of Ba advanced by (Nonaka & Konno, 1998; Nonaka & Toyama, 2004). The shared context created by establishing a project based team including not only software specialists, but representatives of the customer organization as well, achieves the condition of originating Ba by bringing them

into close contact with a shared purpose. Charged with the creation of a product, the communicative and collaborative aspects both within the Agile team and across traditional boundaries into the sponsoring organization create an environment in which the flow of knowledge between the whole team, as Beck styles it, operates with quality, energy and efficiency in creating an interacting Ba that allows them to not only bring to bear their respective skills and knowledge but to also to impart that knowledge to each other. The idea of the planning game and blitz planning epitomize this exchange of expertise as business knowledge of user or market priorities is balanced against developer knowledge of what it will take to implement various aspects of the proposed features, resulting in new knowledge and understanding between the parties and the opportunity to achieve greater success through creative solutions and negotiation. The cyber Ba is represented in the completed iteration and its delivery to the organization, which has the effect of informing it and creating the exercising Ba where this new knowledge transfers back to the individual.

#### Agile Lifecycles and the Knowledge Creation Process

To give further context to the knowledge creation process within the Agile methods we can look at the life cycle models of XP, as shown in (Figure 6) and Crystal represented in (Figure 7). It should not come as a surprise that

Figure 6. Extreme Programming Project Model

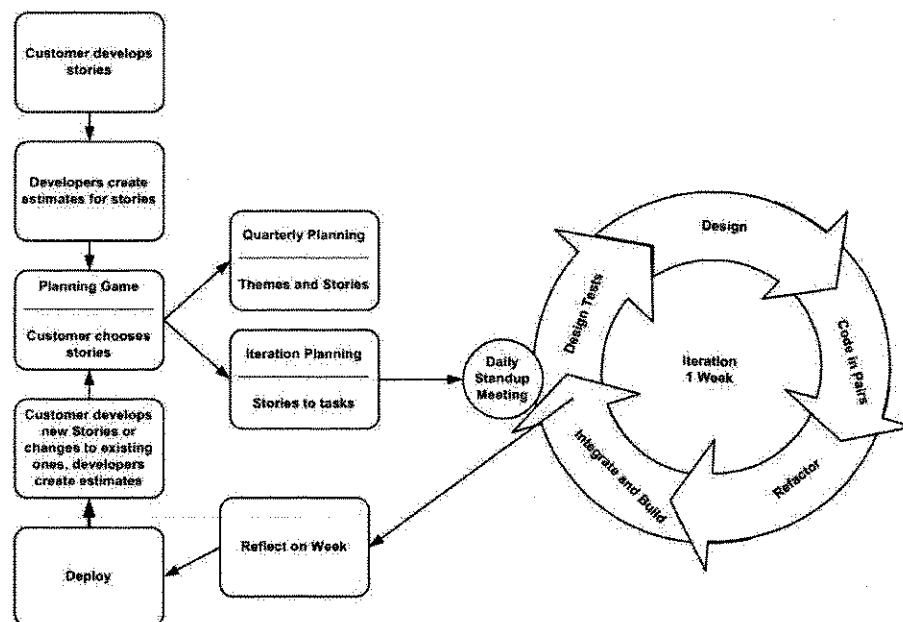
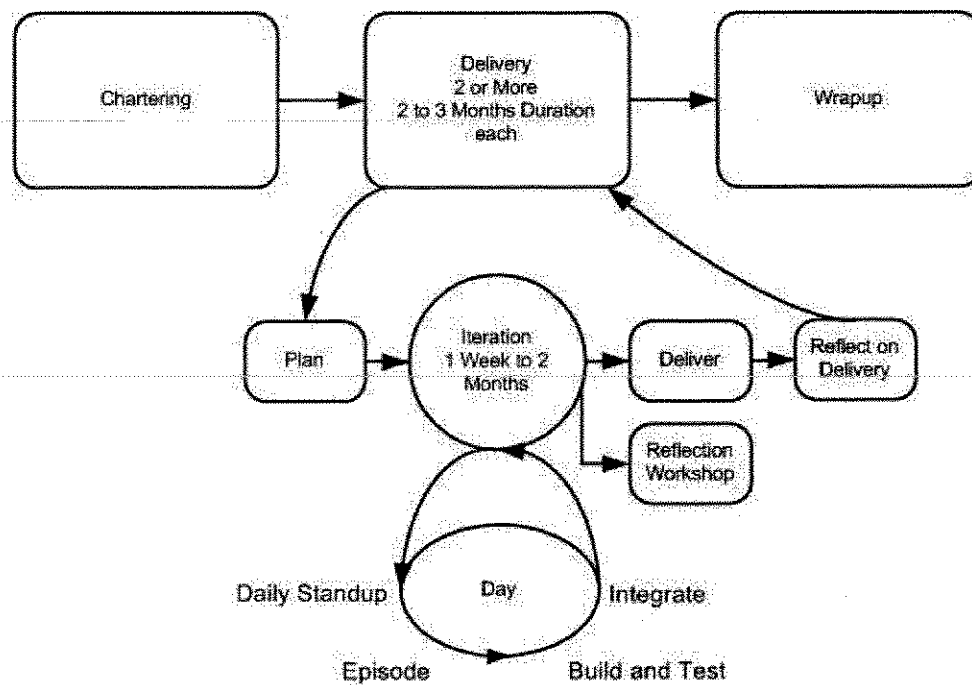


Figure 7. Crystal Clear Project Model



the two models are more similar than different.

The model of Agile lifecycle just presented can be related to the “five-phase model of the organizational knowledge creation process” presented by (Takeuchi & Nonaka, 2004) in the last chapter, taking into account only minor changes to accommodate the specific nature of software development.

The first three phases sharing tacit knowledge, creating concepts and concept justification tend to occur in unison at the beginning of the lifecycles, with the team being assembled, the creation of system metaphors, stories, and preliminary planning taking place. While Crystal makes specific provision for a technical justification of the proposed solution, the planning stages of both XP and Crystal provide an opportunity for the customer to explore their own motivations and priorities related to the proposed system to ensure that they coincide with their needs.

The fourth phase, building an archetype is clearly the concern of software engineering and it is here that both XP and Crystal elaborate most fully their respective approaches to how that is to be carried out.

In the fifth phase, cross leveling of knowledge, the product is delivered to the users who then have the opportunity to review and understand how it suits their purposes.

### Conclusion

In this chapter the Agile methods have been related and evaluated in light of the principles of knowledge management. In applying the theories of Nonaka and Takeuchi and applying them to software development it becomes apparent that there exists a high degree of fluidity in the distinctions that the theory draws

between the SECI process, enablers, barriers and the Ba which can pose significant difficulty in the partitioning of various practices among them. This is to be expected as software engineering is a complex field in and of itself, while knowledge as (Davenport & Prusak, 1998) remind us, enables us to deal with such complexity because of its fluidity and the attendant difficulty of reconciling its constituent elements. The Agile methods exhibit a high degree of knowledge creation activity as expressed by their principles, practices and strategies as they relate not only to the way in which software development is carried out, but also in the manner they seek to reflectively revise themselves in light of the context they find themselves. It seems clear that knowledge management and the knowledge creation process play an important role in fostering and grounding the Agile methods, while they provide an informative and practical implementation of knowledge management.



## CHAPTER V

### CONCLUSIONS AND RECOMMENDATIONS

#### Introduction

Tying together the various threads which form the body of this thesis has been no easy matter. We began by looking at the complexity that surrounds and infuses the various disciplines of software engineering and the outcomes that result from this sometimes chaotic interaction. Knowledge management was then introduced as having the ability to help us make sense of the complexity, or at least providing us with a way of working through it to better rend success from failure. The Agile methods a relatively new way of creating software exhibit a high degree of knowledge management in their use of communication, reflection, and knowledge creation activities were elaborated. In this part we will return again to the question of complexity in software engineering and assess the potential exhibited by the Agile methods to manage it.

#### Addressing Software Complexity

Many of the problems that were described in the second chapter of this thesis arise due to the way in which software engineering has traditionally approached software development. The Agile methods offer a new approach to software development founded on the idea that people and communication form the core of this activity. In this section the author revisits these problems from the perspective of the Agile methods.

### Requirement Change

Changes to requirements pose a significant threat to the successful implementation of a system, whether these changes arise from deficiencies in the original specification of requirements or due to changes external to the project. Traditional software engineering resists this sort of change as the requirements form the basis of its design and planning activities which will be impacted if change is accommodated. The Agile methods on the other hand both accept change as inevitable and welcome it. The ability to do so is found in the way that software construction is carried out under the Agile methodologies. By adopting a lifecycle model that encourages a short and iterative approach these methods are able to shift direction quickly in response to the shifting needs of the organization in which it operates. By limiting iterations and deliveries to weeks and months instead of years and proceeding from a test driven approach to coding, they are able to incorporate new functionality and revise already implemented features with both speed and confidence.

### Architecture and Design

The problem of design lock-in in traditional software development arises from a need to establish a firm foundation of requirements. Architecture and design share many of the problems described in the requirements change section which precedes this one. The author would further argue that this difficulty runs even deeper in that once the architecture and design have been defined it becomes very difficult to revisit it without the impact being felt throughout the systems structure. The Agile methods deal with this difficulty by

moving decisions as to architecture and design to the point where they are needed to support the implementation of a particular feature and even then limit the creation of any permanent edifice to just what is needed. This approach provides a degree of fluidity to the system that can accommodate changing requirements while evolving the architecture and design of it. The additional benefit of the test driven approach can also be seen here as developers are able to refactor at both the feature and architectural level while ensuring that the system will still operate correctly.

### Life Cycle Models

As mentioned above the iterative nature of the Agile methods is one of the keys to its ability to adapt to change. No less so is the inclusion of constant planning within it. Rather than seeing software development and project management as distinct disciplines the Agile methods bring them together so that they each inform the other of the tradeoffs that are necessary to succeed. The inclusion of reflective practices within the lifecycle of the project as both a way to improve the process and the product is also a benefit as it informs and makes change possible.

### Customer Knowledge of Business Need

The customer at the beginning of a project comes to it with a certain view of what it is they want to gain from it. As the project proceeds these initial assumptions will change. Alongside this each iteration of the project results in new knowledge not only of the product itself but also of what it can do, this necessitates the revisiting of the assumptions at the beginning of the iteration. As

the customer is exposed to the process and receives feedback from the development team and the users new needs will come to light while others will fade. These changes can be accommodated by reprioritizing the feature sets already established, adding to them and removing them where necessary.

#### Customer Involvement and Sponsorship

The degree of customer involvement in the Agile methods is a serious but necessary commitment. Defining and interpreting the requirements as stories, prioritizing and reprioritizing the order in which they will be implemented and acting as a facilitator between the development team and various stakeholders in the project is a difficult role. In the end, however, who else could do it?

#### IT and Business Need

The Agile methods provide a means of responding to the needs of business and the changing environment in which it operates. It also exposes those who function within the IT area to the thinking and needs of the business, rather than simply passing along requests. With a better understanding of the business one would expect IT to become more proactive in its approach to delivering the projects it is charged with and hopefully advancing new ones.

#### Impact on Complexity

The impact of the Agile methods derives from many of its practices. Chief among them is its focus on communication. Communication between the customer and the development team acts to begin the cycle of knowledge creation that flows throughout the process. Communication between members of the development team not only serves to inform them of the content of the

system, but also of its context as well. It further informs them of the way they are performing their activities allowing them to tune their habits to become better at it. Through reflection the teams learn about itself, its process and its product. Complexity in its many forms is dealt with in these conversations and reflections allowing those involved to make sense of it and react accordingly.

### Recommendations

Many of the topic areas within the Agile methods have been studied, the issue of documentation has however not received the same degree of attention as others. The need for artifacts other than source code and tests seems clear both from the perspective of practitioners of software development and those who study both the methods themselves and project based organizations as well. As we have seen tacit knowledge that is not elaborated and captured is lost. Once captured however they become knowledge precursors. In this respect the elaboration of some minimally intrusive way to capture the knowledge that leads to decisions should be further investigated. The author is particularly interested in:

- the development of a strategy to tie together stories, planning decisions, design decisions, comments, code and tests, as well as any other artifacts that could reasonably inform both the process and those with an interest in it
- the application of team blogs to capture the teams thought process in the design decisions it makes

- the use of subscription based RSS feeds for the delivery of information to team members

While researching this thesis it became apparent that the practices of product design could also hold value when dealing with the manner and conduct of software development, further research in this area could also add to our knowledge.

The Agile methods extol the practice of pair programming as a way to enhance programmer knowledge of both their tools (programming language) and the system under construction. Establishing a means to ensure that the rotation of programmers maximizes their exposure to all areas of the system would also be of assistance in this area.

## REFERENCES

- Abran, A., Moore, J. W., Bourque, P., & Dupuis, R. (Eds.). (2004). *Guide to the software engineering body of knowledge*. Los Alamitos, CA: IEEE Computer Society.
- Al-Kilidar, H., Parkin, P., Aurum, A., & Jeffery, R. (2005). *Evaluation of effects of pair work on quality of designs*. Paper presented at the Proceedings of the 2005 Australian Software Engineering Conference (ASWEC'05), Brisbane, Australia.
- Ambler, S. (2005). Agile documentation. Retrieved February 8, 2005, from <http://www.agilemodeling.com/essays/agileDocumentation.htm>
- Argyris, C. (1998). Teaching smart people how to learn. In P. F. Drucker (Ed.), *Harvard Business Review on knowledge management*. Boston, MA: Harvard Business School Press.
- ATKearney. (2005). Why today's IT organization won't work tomorrow. Retrieved October 4, 2005, from [http://www.atkearney.com/shared\\_res/pdf/IT\\_Tomorrow\\_S.pdf](http://www.atkearney.com/shared_res/pdf/IT_Tomorrow_S.pdf)
- Beck, K. (2002). *Test-Driven development by example*. Boston, MA: Addison-Wesley.
- Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change* (2nd ed.). Boston, MA: Addison-Wesley Professional.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001a). Manifesto for agile software development. Retrieved March 19, 2006, from <http://AgileManifesto.org>

- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001b). Principles behind the Agile Manifesto. Retrieved March 19, 2006, from <http://AgileManifesto.org>
- Boehm, B. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61-72.
- Boehm, B. (2000). Project termination doesn't equal project failure. *IEEE Computer*, 33(9), 94-96.
- Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Upper Saddle River, NJ: Addison-Wesley Professional.
- Boehm, B., & Turner, R. (2005). Management challenges to implementing Agile processes in traditional development organizations. *IEEE Software*, 22(5), 30-39.
- Bresnen, M., Edelman, L., Newell, S., Scarbrough, H., & Swan, J. (2005). A community perspective on managing knowledge in project environments. In P. Love, P. Fong & Z. Irani (Eds.), *Management of knowledge in project environments*. Oxford, UK: Butterworth-Heinemann College.
- Canfora, G., Cimitile, A., & Visaggio, C. A. (2004). *Working in pairs as a means for design knowledge building: an empirical study*. Paper presented at the Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC'04), Bari, Italy.
- Carr, N. G. (2003). IT doesn't matter. *Harvard Business Review*, 81(5), 41-48.
- Ceschi, M., Sillitti, A., Succi, G., & De Panfilis, S. (2005). Project management in plan-based and agile companies. *IEEE Software*, 22(3), 21-27.



CHAOS: A recipe for success. (1999). West Yarmouth, MA: The Standish Group International, Inc.

Chau, T., & Maurer, F. (2004). *Knowledge sharing in agile software teams*. Paper presented at the Proceedings Symposium Logic versus Approximation.

Cicmil, S. (2005). Reflection, participation and learning in project environments: A multiple perspective agenda. In P. Love, P. Fong & Z. Irani (Eds.), *Management of knowledge in project environments*. Oxford, UK: Butterworth-Heinemann College.

Cockburn, A. (2004). Agile software development (MP3 Recording). Kentfield, CA: ITConversations. Retrieved December 28, 2005, from <http://www.itconversations.com/shows/detail175.html>

Cockburn, A. (2005). *Crystal clear: A human-powered methodology for small teams*. Upper-Saddle River, NJ: Pearson Education Inc.

Davenport, T. H., & Prusak, L. (1998). *Working knowledge: How organizations manage what they know*. Boston, MA: Harvard Business School Press.

de Souza, S. C. B., Anquetil, N., & de Olivera, K. M. (2005). *A study of the documentation essential to software maintenance*. Paper presented at the Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information, Coventry, UK.

- Dingsøyr, T., & Conradi, R. (2003). Usage of intranet tools for knowledge management in a medium-sized software consulting company. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Drucker, P. F. (1998). *Harvard Business Review on knowledge management*. Boston, MA: Harvard Business School Press.
- Dutoit, A. H., & Paech, B. (2003). Eliciting and maintaining knowledge for requirements evolution. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Dybå, T. (2003). A dynamic model of software engineering knowledge creation. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Edwards, J. S. (2003). Managing software engineers and their knowledge. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Gladwell, M. (2002). *The tipping point*. New York, NY: Back Bay Books.
- Glass, R. L. (2005). IT failure rates--70% or 10-15%? *IEEE Software*, 22(3), 110-112.
- Gustafasson, M., & Wikstrom, K. (2005). Managing projects through reflection. In P. Love, P. Fong & Z. Irani (Eds.), *Management of knowledge in project environments*. Oxford, UK: Butterworth-Heinemann College.
- Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *IEEE Computer*, 34(11), 120-122.

- Hodgetts, P. (2004). *Refactoring the development process: Experiences with the incremental adoption of Agile practices*. Paper presented at the Proceedings of the Agile Development Conference (ADC'04), Salt Lake City, Utah.
- Ichijo, K. (2004). From managing to enabling knowledge. In H. Takeuchi & I. Nonaka (Eds.), *Hitotsubashi on knowledge management*. Singapore: John Wiley & Sons (Asia).
- iTWire. (2004). IT project failures up sharply according to US report. Retrieved October 4, 2005, 2005, from <http://www.beerfiles.com.au/content/view/367/37/>
- Jeffries, R., Anderson, A., & Hendrickson, C. (2000). *Extreme programming installed*. Boston, MA: Addison-Wesley Professional.
- Kelly, T. (2005). *The ten faces of innovation*. New York, NY: Doubleday.
- Kochikar, V. P., & Suresh, J. K. (2003). The Infosys km experience. In M. Rao (Ed.), *Leading with knowledge: Knowledge management practices in global infotech companies*. New Delhi: Tata McGraw-Hill.
- Kroll, P., & Kruchten, P. (2003). *The rational unified process made easy: A practitioners guide to the RUP*. Boston, MA: Addison Wesley.
- 
- Larman, C. (2003). *Agile and iterative development: A manager's guide*. Boston, MA: Addison-Wesley Professional.
- Leonard, D., & Straus, S. (1998). Putting your whole company's brain to work. In P. F. Drucker (Ed.), *Harvard Business Review on knowledge management*. Boston, MA: Harvard Business School Press.

- Lindvall, M., & Rus, I. (2003). Knowledge management for software organizations. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Love, P., Fong, P., & Irani, Z. (2005). *Management of knowledge in project environments*. Oxford, UK: Butterworth-Heinemann College.
- Marchesi, M., Succi, G., Wells, D., & Williams, L. (2002). *Extreme programming perspectives*. Boston, MA: Addison-Wesley Professional.
- Martin, A., Biddle, R., & Noble, J. (2004). *The XP customer role in practice: Three studies*. Paper presented at the Proceedings of the Agile Development Conference (ADC'04), Salt Lake City, Utah.
- Martin, J. (1991). *Rapid application development*. New York, NY: MacMillan Publishing.
- Martin, J., & McClure, C. (1988). *Structured techniques for computing*. Upper Saddle River, NJ: Prentice Hall.
- Melnik, G., & Maurer, F. (2004). *Direct verbal communication as a catalyst of Agile knowledge sharing*. Paper presented at the Proceedings of the Agile Development Conference (ADC'04), Salt Lake City, Utah.
- 
- Newell, S., & Huang, J. (2005). Knowledge integration processes and dynamics within the context of cross-functional projects. In P. Love, P. Fong & Z. Irani (Eds.), *Management of knowledge in project environments*. Oxford, UK: Butterworth-Heinemann College.

- Nonaka, I. (1998). The knowledge creating company. In P. F. Drucker (Ed.), *Harvard Business Review on knowledge management*. Boston, MA: Harvard Business School Press.
- Nonaka, I., & Konno, N. (1998). The concept of "ba". *California Management Review*, 40(3), 40 - 53.
- Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. New York, NY: Oxford University Press.
- Nonaka, I., & Toyama, R. (2004). Knowledge creation as a synthesizing process. In H. Takeuchi & I. Nonaka (Eds.), *Hitotsubashi on knowledge management*. Singapore: John Wiley & Sons (Asia).
- Polanyi, M. (1962). *Personal knowledge*. Chicago, IL: The University of Chicago Press.
- Polanyi, M. (1967). *The tacit dimension*. New York, NY: Anchor Books.
- Quinn, J. B., Anderson, P., & Finkelstein, S. (1998). Managing professional intellect. In P. F. Drucker (Ed.), *Harvard Business Review on knowledge management*. Boston, MA: Harvard Business School Press.
- 
- Rao, M. (2003). *Leading with knowledge: Knowledge management practices in global infotech companies*. New Delhi: Tata McGraw-Hill.
- Raskin, J. (2005). Comments are more important than code. Retrieved April 27, 2005, from <http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=290>

- Royce, W. W. (1987). *Managing the development of large software systems: Concepts and techniques*. Paper presented at the Proceedings of the 9th International Conference on Software Engineering, Sevastapole, CA.
- Schach, S. R. (2002). *Object-oriented and classical software engineering* (5th ed.). New York, NY: Mcgraw-Hill.
- Schwaber, K., & Beedle, M. (2001). *Agile project management with scrum*. Redmond, WA: Microsoft Press.
- Schwalbe, K. (2002). *Information technology project management*. Boston, MA: Course Technology.
- The Standish Group Report - CHAOS. (1995). West Yarmouth, MA: The Standish Group International, Inc.
- Takeuchi, H., & Nonaka, I. (2004). *Hitotsubashi on knowledge management*. Singapore: John Wiley & Sons (Asia).
- Tiwana, A. (2002). *The knowledge management toolkit*. Upper Saddle River, NJ: Prentice Hall PTR.
- Verner, J. M., & Evancho, W. M. (2003). An investigation into software development process knowledge. In A. Aurum (Ed.), *Managing software engineering knowledge*. New York, NY: Springer-Verlag.
- Wake, W. C. (2001). *Extreme programming explored*. Boston, MA: Addison-Wesley Professional.

Williams, L., Shukla, A., & Antón, A. I. (2004). *An initial exploration of the relationship between pair programming and Brooks' law*. Paper presented at the Proceedings of the Agile Development Conference (ADC'04), Salt Lake City, Utah.

## APPENDIX A

### PRINCIPLES BEHIND THE AGILE MANIFESTO

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress. Agile processes promote sustainable development.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

(Beck et al., 2001b)