ADOPTING AGILE SYSTEM DEVELOPMENT METHODOLOGIES

Eugenia Y. Huang
Dept. of MIS, National Chengchi University, Taipei, Taiwan
huang.eugenia@gmail.com, +886-2-29387348 ext. 81225
Sheng-Wei Lin
Dept. of CSIM, Soochow University, Taipei, Taiwan
swlin@csim.scu.edu.tw, +886-2-23111531 ext. 3810
John Chinhan Tang
IBM Taiwan Corporation, Taipei, Taiwan
soup5678@gmail.com, +886-2-87239771

ABSTRACT

Awareness of agile system development methodologies (SDM) has grown among information system development community in recent years. Many advocates stress the merits of agile SDM's flexibility which seems to go well with the trend of shortened development cycle. Yet its adoption is critically determined by the software developers. In order to gain the understanding of software developers' adoption intention, this paper conducted an exploratory study to investigate the software developers' intention to adopt agile SDMs. Delphi and AHP methods were applied in tandem to identify the factors that may affect this intention and their relative importance. The result showed that factors of organization dimension, customer dimension, and team dimension were most valued in the adoption consideration. Contrary to our expectation, the effect of individual dimension was minimal. The factors along with their dimension weighting and factor weighting advise IS managers what to avoid and what to emphasize when introducing agile SDM into their organizations.

Keywords: Agile software development, exploratory research, Delphi, AHP.

INTRODUCTION

Information technology made a huge leap in the last decade, and increasingly it plays a key role in enabling business transformation. Ever intensive global competition and round-the-clock business transactions have also sped up the rate of information exchange dramatically. Decades ago when information systems (IS) were first used in business, their main function was routine data processing. Their capability was soon evolved to support firm's internal information integration

and executive decision making. Now IS is widely deemed a valuable resource which when integrated with business processes, creates firm's sustainable competitive advantage (Wade & Hulland, 2004).

System development methodologies (SDMs) were born in the early sixties in response to the software crisis of budget overrun and schedule slip. When the emphasis of software development shifted from piecemeal functionality to overall quality, only systematic development methodologies can assure project success. Plan-driven methodologies were thus created and gradually improved and evolved to accommodate business demand in software development for nearly thirty years. The impact of rapidly changing business environment, difficulty in assessing the market demand, constantly changing customer requirements, and short project delivery time also drive the evolution of system development techniques (Baskerville et al., 2001).

In the early history of IS development, most IS projects were internally designed and implemented with armies of software developers at data center and lengthy planning and review stages. Now decades later, firms demand rapid development with fewer developers. Fitzgerald (1998)'s survey showed that the average project duration had decreased to six months and the average numbers of developers also decreased to three. These were some of the key changes that mandated the necessity of IS outsourcing. Surveys showed that high percentage of firms had outsourced or purchased off-the-shelf packages (Fitzgerald, 1998; Necco et al., 1987), and increasingly less organizations developed the system solely internally. With the remarkable shift toward outsourcing alternatives and the trend of rapid development, the percentage of development teams using SDMs was also declining. Fitzgerald thought that it was attributable to the lack of suitable SDMs.

The need for newer SDMs that depart from the traditional plan-driven principle became apparent and in the late nineties a category of agile SDMs emerged. Agile SDMs focus on feedback mechanism instead of prior planning, and on human interactions instead of procedures. These methodologies were intended to enhance customer response and market feedback, so as to increase firms' capability of outpacing their competitors.

The benefits of agile SDMs in the fast-paced context seem to be self-evident. In addition, with more than a decade of fine tuning, agile SDMs have developed from drawing boards that were full of concepts to rich sets of tools that can support software development. Yet, the adoption of SDMs has never picked up. Although researches on SDMs exist, gaps remain. Currently, there is limited research on the adoption of agile SDMs. Therefore, the purpose of this paper is to explore software developers' intention to adopt agile SDMs by understanding the scenarios of adopting or not

adopting agile SDMs. To address the issues described above and begin to fill the gaps in the extant research, the present study was designed to address the following research questions: What factors affect software developers' intention to adopt or not to adopt agile SDMs? The purpose of this research is thus to understand the backdrop of the emergence of agile SDM through literature review, in order to assess the fit with specific contexts.

To investigate the key factors that influence software developers' intention to adopt or not to adopt agile SDMs. The key factors were identified by using Delphi and Analytic hierarchy process (AHP) methods, as described in the Research Design section. This study is structured as follows: Section 2 provides an introduction to the emergence and evolution of SDMs, including agile SDMs. Section 3 describes the Delphi and AHP methods and Section 4 presents the analysis results of these two methods. Then conclusions and the discussion of implications are presented.

THEORETICAL BACKGROUND AND HYPOTHESES

The emergence and evolution of SDMs

An SDM is a set of best practices involved in the process of designing, building, implementing, and maintaining an information system to provide stakeholders a means to accomplish and manage these activities, to provide users values, as well as to achieve business goals. Its framework usually allows the flexibility of accommodating appropriate practices for special situations (Bentley et al., 2007; Brinkkemper, 1996; Laudon & Laudon, 2002; Roberts et al., 1998).

SDMs are derived from practical experiences or theoretical concepts. The first type of SDMs was derived from consulting firms' practical experiences. These SDMs were the result of consulting firms' effort in commercializing the techniques they developed through accumulative experiences of consulting projects. These SDMs were usually fragmented or informal compared to a theoretically derived methodology, a consequence of consulting firms' approach of enlarging the usage phases or adding techniques to existing SDMs.

The second type of SDMs is derived from theoretical concepts, which are brought out by universities or research centers, and apt to be hard to commercialize. Usually this type of SDMs are influential in concepts and could stimulate ideas, which is a result of the philosophical meaning of improving the SDMs to build success systems (Avison & Fitzgerald, 1999, 2003).

A system is delivered by internal and/or external team projects. However, managing a successful software system development project requires different team project management methodologies.

In general, the stages of SDMs include planning, analyzing, designing, implementing, manipulating, maintaining, controlling, auditing, while project management methodologies include planning, analyzing, designing, execution, evaluation, correction and completion. When managing a software project, many managers tend to apply the general project management methodologies they are familiar with to software development. A main theme of general project management methodologies includes an assessment of resource allocation and control. Although software project has all the traits of a general project, its uniqueness requires a shift of focus to the process and dynamics of developing a system.

SDMs are generally categorized by the nature of development process and dynamics. There are waterfall-based SDMs, interactive-based SDMs, and agile-based SDMs. These three categories of methodologies emerged in different eras with unique development scenarios.

From sixties to early seventies, there were no explicit or formal methods in developing information systems. In most circumstances, major issues were found in technical and programming aspects, and particularly resulting from computer hardware. Thus in pre-methodology era, developers focused on solving technical problems in isolated scopes. This resulted in poor management and control of the system. Facing the challenges of management and control, a more disciplined approach to develop information systems was gradually seen necessary (Avison & Fitzgerald, 1998).

During late seventies to early eighties, discussions by developers to improve corresponding principles, tools, documentations and manageable rules increased. The first formulated approach that has come to be known as Systems Development Life Cycle (SDLC), or, more commonly, the waterfall model. The model was proposed by Royce (1987). In the beginning it was only a conceptual model which suggested a formal sequence for developers to follow. It focused on segmenting development processes into various phases, and emphasized that each phase had to be completed before the next one could start, and each phase had a set of defined outputs or deliverables to be produced before it could be considered complete.

Various formal documentations mark the end of each phase and benefit the communication between users and developers, and aid maintenance and training. Furthermore, checkpoints are built in along the stages, which offer systematic ways to help assuring timely attainment of milestones. These traits effectively solved many problems of pre-methodology era (Avison & Fitzgerald, 1999; Royce, 1987).

However, some shortcomings of SDLCs were gradually observed: 1. User demand is unstable under rapid changing business environment. 2. SDLC leads to rigidity in design, because each stage is driven by the result of previous stage, thus a change in any phase would have a chain effect on the forwarding phases. 3. Users are not satisfied with technical-based documents which are hard to understand as a user (Avison & Fitzgerald, 1998; Sommerville, 1996).

The eighties marked a climax of the number of models proposed concerning SDMs; such as Structured Systems Analysis and Design Method (SSADM), Iterative and incremental development, data oriented, prototyping, object-oriented, participative development, strategic development and iterative models (Avison & Fitzgerald, 1999; Sommerville, 1996). These models are not mutually exclusive, and sometimes more than one model can be referenced in an IS project. However, all models aimed to solve the problems of the original waterfall methodology, with each model offering a correction to SDLC's problems, and together they took a major departure from SDLC.

But the environment of the nineties underwent a revolutionary change, due to the commercialization of Internet, rapid advancement of Internet technology, and the high-level interconnectivity of the networking infrastructure (Brancheau et al., 1996). The Internet accelerated the market operation and shortened the time to market. Such environment shaped the emphasis of efficient delivery of information systems (Brancheau et al., 1996). Furthermore, waterfall model was originally designed for large-scale system development which usually was not characterized by fast turn-around. Using the same waterfall model in a rapidly changing market may lack the necessary agility and flexibility (Avison & Fitzgerald, 2003).

In the late nineties, new SDMs such as Scrum, extreme programming, adaptive software development, feature-driven development, dynamic system development method, and etc, are proposed. In 2001, seventeen experienced software developers, some of them authors of these new developments, published "Agile Manifesto" (Highsmith & Cockburn, 2001). In the manifesto, they described that in their effort to uncover better ways of developing software and to help others to do the same, they came 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.

Nowadays SDMs which adhere to the principles of Agile Manifesto are called agile SDMs. Conboy's review of literature (2009) on agility found that agility was not new in concept and was

mentioned in the mainstream of business articles in the early nineties. In these articles agility is related to terms such as flexibility and leanness, and is used in management, manufacturing, organization behavior and other disciplines. When the concept of agility is applied to system development, the flexibility of system development is to feedback rapidly, commit to innovate and proactive response, and learn from change. And the concept of leanness is to learn continuously and thus contribute to creating value for customers, which may be in value of economic, quality, or simplistic.

In the system development context, agility means adapting to differences in different projects. The methods and procedures used are allowed to change under the concept of agility, aiming to synchronize the development and feedback rapidly with the customers, in order to create values for customers, which may be economic, quality, or simplistic. This practice is important, as customer needs are constantly changing throughout the development process. Some may worry that without a rigid procedure, the result of development might be unstable or incomplete. Thus in agile SDMs the concept of iterative and incremental development is brought out to help insure stability and completeness (Cockburn, 2002).

Because agile SDMs also emphasized rapid feedback, each sub-module must be completed in the allotted time. During the allotted period the needs of the parent module will not change until the completion of the entire sub-module. Finally from bottom up all modules are integrated to complete the development. Similar to prototyping, rapid feedback achieves the purpose of confirming user demands. But prototyping not necessarily delivers the actual subsystem, while the product of fast feedback is actual sub-system.

Joint development with users is valued in agile SDMs; it facilitates the shared experience communication. Through this type of communication, users and developers are regularly confirming the demands, discussing difficulties encountered, adjusting the schedule.

The birth of Agile SDMs was in response to the dynamic business world (Beck, 1999). But a decade after Agile Manifesto was published, the usage of agile SDMs is still low. Some regarded agile SDMs as experiential methodologies, and when firms use SDMs in IS development, they often made use of only partial aspect of the methodology. (Fitzgerald et al., 2006). Boehm (2002) believed that knowing in which situation to use agile SDMs is important. He viewed agile SDMs and traditional SDMs as two poles of a spectrum, and suggested that a team should identify its position along the spectrum before choosing a specific SDM.

Apparently, the spectrum Boehm was referring to could not be a simplistic one-dimensional axis. There are numerous factors to consider, for example, whether the characteristics of individual developers and the development team could benefit from agile SDMs (Boehm, 2002; Conboy, 2009). Agile SDMs provide developers with autonomy with regard to how tasks are accomplished, thus if developers are lack of self-regulation, project schedule will be delayed. Moreover, shared understanding is necessary for a team to collaborate and benefit from agile SDMs. Further, working habits of teammates and users determine whether ongoing feedback stressed by agile SDMs can be carried out (Harris et al., 2009). Ongoing feedback among team members helps dynamic task appropriation and mutual learning, which are central to agile SDMs. Ongoing feedback between the team and users can be more challenging, as users' willingness and commitment to the project are not guaranteed. Boehm (2002) and Harris (2009) pointed out that if the specification of the system was clear and not subject to change, plan-driven SDMs would be a better option. On the contrary, when users were not sure about the specifications or when the system was facing a fluctuating market, then choice has to be agile SDMs. A team needs to be structured and performance indicators designed to make the team work as a group and support each member to set his goals and then self-regulate and self-monitor the progress in achieving the goals (Harris et al., 2009). If such structure and performance evaluation scheme are in place, the outcome control is adequate for the project to benefit from agile SDMs. Thus, characteristic of developers, ongoing feedback, system scope, and outcome control are the factors that a team needs to consider before using agile SDMs.

RESEARCH METHOD

The objective of this study is to identify factors that may affect software developers' intention to adopt agile SDMs, and to rank the relative importance of these factors. As extant research offers limited understanding with regard to this objective, a Delphi survey serves a readily available method to explore the possible factors and to determine the preliminary ranking of them. However, the Delphi survey procedure can only rank these factors as individual entities and is incapable of performing two-layer ranking. Missing out this perspective can be a major loss of information because a two-layer structure is essential for presenting a holistic view. Therefore, following the Delphi survey, AHP method was applied to calculate the weightings of each dimension and each factor.

Combined Delphi-AHP approach

A combination of Delphi and AHP methods has been used in the context of group decision for various research purposes. Some studies (Tavana et al., 1996; Banuls & Salmeron, 2007) used the

Delphi results as the input to AHP method, while others (Zhanga et al., 2004; Tsaia & Sub, 2005) applied each method at different phases of decision-making process. Our study took upon the former approach, i.e. using Delphi and AHP in tandem.

The Delphi Survey

The Delphi method is used to find the reliable consensus of expert opinion group (Dalkey & Helmer, 1963), which in this study is a group of new generation of software developers. Since agile SDMs are relatively new and the usage is low, many of the older generation of software developers may not be familiar with or even know about agile SDMs. Therefore the focus of our inquiry should be on the new generation of software developers, who were more likely to be taught about agile SDMs as an IS student or as a young IS professional.

Recruiting the participants

Delphi survey calls for participants who have a deep understanding of the topic being studied (Okoli & Pawlowski, 2004). All participants in this study had experience with a number of IS projects. In addition, they were new generation of software developers and had training with agile SDMs, as opposed to the developers of the previous generation who had no knowledge of agile SDMs. This choice of sample helps to ensure that the findings possess general validity.

The AHP method

The AHP method is useful in evaluating criteria and developing priorities for goals in uncertain decision-making condition (Saaty, 1980). The AHP method decomposes a complex decision problem into a hierarchical structure and constructs a pair-wise comparison matrix to weight goals and its subsequent alternatives in each hierarchy level, which could avoid individual's subjective opinion biasing the results. Higher weighting scores correspond to higher priority goals or alternatives.

RESULTS

The participants, 18 male and 3 female, aged 24 to 35, who volunteered to take part in the study were all information system professionals. Eight of them were system analysts, nine were programmers and four were project managers. They came to know about agile SDMs either at the capacity of an IS student or as an on-job training.

Findings from the Delphi survey

There are a few interesting observations of the pattern of change. First, the factors in the individual dimension disappeared all together in the third round, which might indicate that developers gradually dropped the consideration of factors in the individual dimension through learning from peer communication, as in the Delphi survey process. Second, the factors of organization dimension, e.g., top manager's support, and user dimension, e.g., user attitude, were stabilized through the iterations of the third round, which indicated that these factors were consistently important. Third, as the number of factors in other dimensions decreased, the percentage of factors in team dimension increased and the team dimension emerged to be the dimension with most factors. It was clearly that software developers' main consideration of whether to adopt agile SDMs were greatly affected by how they perceive their team members' personality, work habit and mutual understanding. Fourth, the number of factors in methodology dimension decreased most dramatically (from 19 to 1 for the adoption case and from 20 to 0 for the non-adoption case), which might mean that peer learning through communications and feedbacks were very effective in toning developers' favoritism attitude toward agile SDMs. Finally, although the factors in success case dimension were very small (2 only) to start with, at the end of the third round this dimension did not disappear. Therefore, even with the lowest percentage in terms of number of factors, success case is an indispensable dimension.

At the closure of the Delphi survey, six dimensions of adoption scenario and four dimensions of non-adoption scenario were identified, with ten factors in each scenario. These results were the inputs to AHP analysis. Overlaps between two scenarios existed, because it is understandable that the converse of the reason for adopting might likely be the reason for not-adopting. However, one-to-one correspondence is by no means a definite consequence; this is the source for factors that are unique to each scenario. In fact, only three factors exhibit the reciprocal one-to-one correspondence. In specific, top manager's support was the reason for adoption, and low level of top management support was the reason for non-adoption; team members' excellent communication skills was one factor for adoption, while their poor communication skills was one factor for non-adoption; users' willingness to interact closely caused the intention to adopt, while user's not being able to engage in close interaction prompted non-adoption.

For the non-adoption scenario, five out of ten factors fall in the dimension of team. It strongly pointed out many inhibitors for adoption were concerns about team members' attitude, ability, habit, team culture, etc.

Two dimensions were present in the adopting scenario but absent in the not-adopting scenario; they are project and methodology dimensions. Extant research usually approached the problem of not adopting or low percentage of use by analyzing the fit between project and methodology, and claimed that agile SDMs were only suitable for certain types of projects. Our not-adopting scenario surprisingly excluded all factors in these two dimensions. This implied that a not-adopting intention probably had to do with neither the nature of the methodology nor the characteristics of the project.

Findings from the AHP analysis

In terms of the adoption scenario, Table 1 showed that user's need to see milestones, top manager's support, and external success cases are most influential in propelling the intention to adopt agile SDMs. Table 2 showed that the conditions of organization, team, and user dimensions had the greatest influence on the intention not to adopt agile SDMs, because the corresponding weights of these three dimensions were much higher than that of success case dimension. But if we look at the weights of individual factors, the factor of internal failure case (belong to success case dimension) was almost as crucial as low top manager's support (belong to organization dimension), and the next factor in line was "insufficient training before use".

It seemed that for the developers not to resist adopting agile SDMs, IS managers must secure top mangers' support, organize adequate training to get developers prepared, and engage carefully so as not to create internal failure case. If developers' resist was not a problem, then to move onto the phase of developers' buy-in, convincing the developers of users' need to see milestones, assuring them of top management's support, and using external success cases as examples were most important tasks for IS managers.

Table 1. Ranking of factors – Adoption Scenario

Dimension (weight)	Factor	Weight	Ranking
Organization (16.2%)	Top manager's support	16.2%	2
Team (22.9%)	Collaboration inclination	8.8%	5
	Close communication	7.2%	8
	Excellent communication skills	6.9%	10
Project (17.2%)	Tight project schedule	7.5%	7
	Dynamic requirement	9.4%	4
User (25.6%)	Need to see milestones	18.4%	1
	Close interaction	7.2%	8
Methodology (7.8%)	Agile enables tight schedule control	7.8%	6
Success Cases (10.3%)	External success cases	10.3%	3

Table 2. Ranking of factors – Non-Adoption Scenario

Dimension (weight)	Factor	Weight	Ranking
Organization (32.0%)	Insufficient training before use	13.7%	3
	Low top management support	18.3%	1
Team (29.2%)	Identification with the agile SDM	9.8%	5
	Poor communication skills	4.1%	10
	Peers preferred worked alone	5.1%	8
	Rigid team culture	5.6%	7
	Communication cost increases	4.8%	9
User (22.2%)	Inadequate IS literacy	9.5%	6
	Cannot afford close interaction	12.7%	4
Success Cases (16.7%)	Internal failure cases	16.7%	2

CONCLUSIONS

In an attempt to understand the low usage of agile SDMs, this research conducted a Delphi survey followed by AHP analysis to identify the important factors that affect the software developers' intention to adopt or not to adopt agile SDMs. With these two methods applied in tandem, not only the important factors were discovered, but also the dimensions and individual factors were ranked by calculating the corresponding weights. These two lists of factors along with their dimension weighting and factor weighting provide IS managers the reference to introduce agile SDM into their organization. In specific, they provided comprehensive checklists advising IS managers what to avoid and what to emphasize.

The consideration of the adoption and non-adoption scenarios separately was crucial, as indicated by the result that only three factors "co-existed" in both lists of factors. Common sense seemed to imply that the reciprocal of the factor for adoption was naturally the factor for non-adoption; for example, users' willingness to interact closely was a factor for adoption, and its reciprocal, "users not committed to close interaction" was a factor for non-adoption. Factors for non-adoption are like inhibitors, and factors for adoption are like enablers. Identifying inhibitors are equally important as identifying enablers, if not more important, because for enablers to be in full action, the inhibitors had better be absent.

The inhibitors were the factors in organization, team, user, and success case dimensions, and the enablers were the factors in organization, team, project, user, methodology, and success dimensions. Surprisingly, there was only one factor in the methodology dimension, out of a total of 20 factors; it was "Agile SDMs enables tight schedule control." It was quite ironic that as we studied issues on the methodology, only 1 out of 20 factors had something to do with the

methodology itself. This seemed to confirm that adopting agile SDMs was not a technical issue; rather, it was more likely a team management issue that has a familiar ring of change management.

For the developers not to resist adopting agile SDMs, IS managers' best strategy would be to secure top mangers' support, organize adequate training, and engage carefully to create internal success case, because internal failure case was weighed heavily by developers. When the resist factors were contained, IS manager needs to exercise effective communications in convincing the developers of users' need to see milestones, assuring them of top management's support, and using external success cases to persuade them.

The analysis of non-adoption scenario resulted in only four dimensions instead of six as in the adoption scenario. Two seemingly important dimensions, project and methodology, were entirely absent from the non-adoption scenario. This result challenges the dominant project-methodology fit perspective of the extant research on agile SDMs. Our result implied that a non-adoption intention probably had little to do with the fit between project and methodology. It does not mean that project-methodology fit should not be in IS managers' mind in choosing a suitable methodology, but it indeed means that this issue does not affect developers' non-adoption consideration.

One interesting result was that all the factors in the final lists described developers' viewpoint looking outwards, and none of the factors reflect developers' opinions on themselves. In fact, in the beginning of Delphi survey, the individual dimension contained the factors about developers' view on self, but all factors were eliminated through the process. This sent out a clear message that developers believed that they themselves were actually fine in accepting the idea of adopting agile SDMs, if there was anything in the way, it had to be other people (e.g., users, teammates), or the environment or context (e.g. support, training, culture, success, etc.)

References

Avison, D. E., & Fitzgerald, G. (1998). *Information systems development: Methodologies, techniques, and tools*: McGraw-Hill Higher Education.

Avison, D. B., & Fitzgerald, G. (2003). Where now for development methodologies? *Communications of the ACM, 46*(1), 78-82.

Banuls, V. A., & Salmeron, J. L. (2007). A scenario-based assessment model- SBAM. *Technological Forecasting and social Change, 74*(6), 750-762.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., & Slaughter, S. (2001). How Internet software companies negotiate quality. *Computer, 34*(5), 51-57.

Beck, K. (1999). Embracing Change with Extreme Programming. *Computer, 32*(10), 70-77.

Bentley, L. D., Whitten, J. L., & Randolph, G. (2007). *Systems analysis and design for the global enterprise*. Boston: McGraw-Hill Irwin.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer, 35*(1), 64-69.

Brancheau, J. C., Janz, B. D., & Wetherbe, J. C. (1996). Key issues in information systems management: 1994-95 SIM Delphi results. *MIS Quarterly, 20*(2), 225-242.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology, 38*(4), 275-280.

Cockburn, A. (2002). Agile software development: Addison-Wesley Longman Publishing Co., Inc.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research, 20*(3), 329-354.

Dalkey, N. & Helmer, O. (1963). An experimental application of the Delphi method to the use of experts. *Management Science, 9(*3), 458-467.

Fitzgerald, B. (1998). An empirical investigation into the adoption of systems development methodologies. *Information & Management, 34*(6), 317-328.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customizing agile methods to software practices at Intel Shannon. *European Journal of Information System, 15*(2), 200-213.

Harris, M. L., Collins, R. W., & Hevner, A. R. (2009). Control of flexible software development under uncertainty. *Information System Research, 20*(3), 400-419.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer, 34*(9), 120-127.

Laudon, K. C., & Laudon, J. P. (2002). *Management information systems: Managing the digital firm*. Upper Saddle River, N.J.: Prentice Hall.

Necco, C. R., Gordon, C. L., & Tsai, N. W. (1987). Systems Analysis and Design: Current Practices. *MIS Quarterly, 11*(4), 461-476.

Okoli, C. & Pawlowski, S. D. (2004). The Delphi method as a research tool: An example, design consideration and applications. *Information & Management, 42*(1), 15–29.

Roberts, T. L., Gibson, M. L., Fields, K. T., & Rainer, R. K. (1998). Factors that impact implementing a system development methodology. *IEEE Transactions on Software Engineering, 24*(8), 640-649.

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering.

Saaty, T. L. (1980). *The Analytic Hierarchy Process* New York: McGraw-Hill.

Tavana, M., Kennedy, D. T., & Joglekar, P. (1996). A group decision support framework for consensus ranking of technical manager candidates. *Omega, 24*(5), 523-538.

Tsaia, M., and Sub, C. (2005). Political risk assessment of five East Asian ports – The viewpoints of global carriers. *Marine Policy, 29*, 291-298.

Wade, M., & Hulland, J. (2004). Review: The resource based vriew and information systems research: Review, extension, and suggestions for future research. *MIS Quarterly, 28*(1), 107-142.

Zhanga, B., Zhanga, Y., Chenb, D., Whiteb, R. E., & Lib, Y. (2004). A quantitative evaluation system of soil productivity for intensive agriculture in China. *Geoderma, 123*, 319-331.