Dakota State University
Beadle Scholar

Research & Publications | College of Business and Information Systems

1-2015

Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case

Yousra Abdo Harb
Dakota State University

Cherie Noteboom
Dakota State University

Surendra Sarnikar
Dakota State University

Follow this and additional works at: https://scholar.dsu.edu/bispapers

Recommended Citation

Harb, Yousra Abdo; Noteboom, Cherie; and Sarnikar, Surendra (2015) "Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case," Journal of the Midwest Association for Information Systems (JMWAIS): Vol. 1 : Iss. 1 , Article 4. Available at: http://aisel.aisnet.org/jmwais/vol1/iss1/4

This Article is brought to you for free and open access by the College of Business and Information Systems at Beadle Scholar. It has been accepted for inclusion in Research & Publications by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.

Journal of the Midwest Association for Information Systems (JMWAIS)

Volume 1
Issue 1 *Volume 2015, Issue 1, January 2015*

Article 4

2015

Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case

Yousra Abdo Harb
Dakota State University, yaharb@pluto.dsu.edu

Cherie Noteboom
Dakota State University, Chrie.Noteboom@dsu.edu

Surendra Sarnikar
Dakota State University, ssarnikar@outlook.com

Follow this and additional works at: http://aisel.aisnet.org/jmwais

Recommended Citation
Harb, Yousra Abdo; Noteboom, Cherie; and Sarnikar, Surendra (2015) "Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case," *Journal of the Midwest Association for Information Systems (JMWAIS)*: Vol. 1 : Iss. 1 , Article 4.
Available at: http://aisel.aisnet.org/jmwais/vol1/iss1/4

This material is brought to you by the Journals at AIS Electronic Library (AISeL). It has been accepted for inclusion in Journal of the Midwest Association for Information Systems (JMWAIS) by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Journal of the Midwest Association for Information Systems

Volume 2015 | Issue 1 Article 3

Date: 10-28-2014

Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case

Yousra Harb
Cherie Noteboom
Surendra Sarnikar

Dakota State University
Madison, SD, 5742 USA
yaharb@pluto.dsu.edu, Cherie.Noteboom@dsu.edu
surendra.sarnikar@dsu.edu

Abstract

Agile methods have attracted significant attention in the industry as an approach for software development and IT project management due to fast-changing business environments, cost, and competitive pressures. Choosing the right approach among various agile development models, however, is a complex, multi-criteria decision that can have significant implications on project success. In this article, we present a teaching case designed to help Information Systems students improve their skills in understanding and evaluating complex business requirements and in selecting the most appropriate software development methodology to match the needs of a specific IT project, and the organization. The teaching case includes a comparative overview of various agile methodologies, as well as the use of multi-criteria decision tools for solving the problem of methodology selection.

Keywords: Teaching case, Software development methodologies, Agile methods, Multi-criteria decision problem

Copyright © 2015 by Y. Harb, C. Noteboom and S. Sarnikar

1. Introduction

Organizations developing custom software projects face the challenging task of selecting the most appropriate software development methodology. IT managers need to identify and select from among several different models and variations of system development methodologies to match the needs of the specific IT project, and the organization. The need for producing better quality, more cost-effective, and faster software solutions has led an increasing number of organizations to adopt agile methodologies in software development (Benefield, 2008). Agile methods in software development have attracted significant attention, given the ever-changing business environment and cost and competitive pressures (Duka, 2013). Over the past decade, several industry surveys have assessed the status of agile method adoption in software development organizations. Project Management Institute (2014) offers a certificate in PMI agile due to the "growing demand in organizations for an agile approach to project management". A "2014 Agile Adoption Mini-Survey" conducted by Ambysoft reports the current level of agile adoption. According to this survey, 33% of respondents said they work in organizations that have succeeded at agile; 11% work in organizations that have had great success at agile; and 11% work in organizations that have tried agile, but it is too early to gauge results[1].

There are many variations in agile development for different types of projects. The most appropriate agile development model or variation depends on an organization's structure and culture (Nerur, Mahapatra, & Mangalaraj, 2005; Vijayasarathy & Turk, 2008); the team's background (Boehm, 2002; Cockburn & Highsmith, 2001); and the set of specific project properties (Chow & Cao, 2008; Hajjdiab & Taleb, 2011). Choosing the right system development methodology calls for careful consideration.

Several aspects of an organization, including its culture, structure, and management practices can have an impact on the successful use of an agile development method (Nerur et al., 2005). Organizational culture exerts a significant impact on an organization's structure, people behavior, decision-making, practices, innovation, and problem-solving methods. Not surprisingly, changing culture is more challenging and time-consuming than changing strategy, structure, or procedures (Adler & Shenhar, 1990). For many organizations, this makes the movement to agile methodologies more difficult (Nerur et al., 2005). In contrast to traditional development approaches, agile methodologies rely more on team work. Thus, effective communication and collaboration among team members are critical for successful adoption of agile methodologies (Cockburn & Highsmith, 2001). A project's nature, the type of project, and its schedule are other important factors that have an impact on the success of agile methods (Chow & Cao, 2008). Understanding the broad aspects of an organization that can impact a software development project is a crucial step in planning and managing the processes of agile development.

Accordingly, learning to analyze business conditions and project characteristics to choose the most appropriate agile methodology is an important skill for students major in Information Systems. This article presents a case study resource for teaching the selection of the most appropriate agile software-development methodology. According to Wei, Xin, and Ying (2010), business case studies play a key role in students' learning and in their transition to the workplace. The case study in this paper gives students an opportunity to learn different agile methodologies, along with a decision-making approach for choosing the most appropriate agile method. The case is developed based on an extensive review of relevant literature and industry surveys on business objectives and factors to be considered when choosing system development methodology (Chow & Cao, 2008; Coffin & Lane, 2006; Dillman, 2003; IIBA, 2011; Leau, Loo, Tham, & Tan, 2012; Martina, 2011; Qumer & Henderson-Sellers, 2007; Taya & Gupta, 2011; Williams, 2007).

2. A2Z Computer Equipment Company Case Overview

A2Z Computer Equipment is a large American semiconductor company. Founded in 1960, it is a leading specialty semiconductor company that produces high-performance audio and graphic processors for computers and mobile devices. The company operates in the United States in California, Maine, Utah, and Pennsylvania. In 2012, it had

[1] http://www.ambysoft.com/surveys/agileJanuary2014.html

grown to 4,000 employees and was earning $70 million annually. The company's aim is to provide semiconductor solutions that help its customers achieve success every day.

The company's software development department located in the state of California employs about 100. The system development section employs more than 60, including the division head, project managers, system analysts, developers, software engineers, and quality assurance testing analysts. Most of the requirements for analysis and software applications and programs are developed in-house by the software and hardware development divisions within the company. A2Z employees have from one to 25 years of experience in software development.

2.1 Problem Description

Since the company was founded, staff have used a waterfall approach. The waterfall model focuses on a set of sequential steps and processes to produce an application. Development starts by defining and analyzing the requirements for the software (Duka, 2013). Activities are performed sequentially, step-by-step, from requirements analysis to maintenance (Purcell, 2012). Waterfall is considered a classical model (Holodnik-Janczura & Golinska, 2010), and relies on a predetermined process where all requirements are known before development starts (Leau et al., 2012). Knowledge passes from one specialist to the next, and, in many cases, feedback cycles are long or even nonexistent (Duka, 2013).

With the waterfall-based development process, projects are built through extensive planning, with the emphasis on a formal and long-term design process. Requirements are largely stable, defined and documented early in the project's life cycle, and the resulting specifications are rigidly maintained during subsequent development phases.

Recently, the company has placed more emphasis on team participation and collaboration. Managers also realized that internal customer (user) involvement during solution development mitigates one of the most consistent issues in software projects: "what they will accept at the end of the project differs from what they told us at the beginning." To improve its software development processes, the company hired an external consultant to identify issues with the current process and areas for improvement.

A review of the software development unit by the external consultant uncovered several issues involving the company's software development and delivery process. The waterfall approach the company used resulted in issues such as waste of resources (people and time), long development cycles, mis-communication, buggy and error-prone software, and low internal customer involvement throughout the project. In addition, there was limited customer feedback in deciding the changes to be made and the priorities of features to be added to the next releases. Although each output produced was subject to quality assurance review and approval, audits to approve releases were less frequent. Typically, audits occurred no more than once a year. As a result, all output produced was not audited by quality assurance to ensure compliance with the defined standards and procedures. Due to fast-paced changes in the technology industry, however, the company suffered from mismatches between delivered functionality and end-user expectations. As one company representative pointed out, *"we need feedback from clients, so we can have better understanding."* The development team planned for releases based on a requirements document. They divided the work into tasks over a long project schedule. Using this approach, the analyst reviewed the requirements, the coder coded, the designer designed, and so on. The complexity of each release increased as the customer base grew. More requirements were subject to change or addition during the release schedules. The releases, however, were not meeting changing requirements, and customer expectations, and the timeframes between them were unacceptable. It was noted "*Unfortunately, the clients are not involved during the projects, and the development life cycle is also long*". These facts were seen as a drawback in the rapidly changing market that the company operates in. As a consequence, managers were looking for a better solution to declining morale and difficulty in responding to frequent changes and demands.

2.2 Process Alternatives

Agile methodology has become the choice for many organizations that are aware of the benefits of adopting the method. At A2Z Co. the project manager became aware of agile. The reasons the project manager started thinking about agile methodology was because of its advantages and the drawbacks of the current software process. The project manager conducted an extensive review of agile methods and shared with the project team a detailed overview of various agile methods. This report is included in Section 2.4.

The project manager wanted to determine if agile methods could solve the problems occurring in the current software development process. After discussion with the team members, the project manager and the team decided that agile methods may help them improve their processes. The project manager said that *"agile methods may be a*

good solution for developing software." The system analysts and developers also indicated that *"waterfall might work for others, but agile will be much easier when developing product, and you can see the progress and everything."* As a consequence of having bad experiences with the waterfall approach, the team started believing in agile methodology benefits. They hoped it would help development teams come up with applications collaboratively and see the progress of their projects.

2.3 A New Project Opportunity

The company received a contract from the U.S. Department of Defense for a virtual simulation environment where graphics processors are to be integrated with multiple devices with varying capacities and characteristics, including high-performance computers and mobile devices. As a part of the contract, the company initiated a software development project for simulation control that involves development of custom device drivers, interfaces, and apps. Given the challenges faced by the waterfall approach, managers realized the importance of undertaking extra tasks to insure internal customers collaborate or participate in their project. As a result, a product council was formed, and the product development process was guided by the council, which includes the project manager, customer, quality assurance testing analyst, software developers and quality control (see Table 1).

Title	**Role description**
Project manager	Responsible for the whole project, produces project management plan, ensures that project time, quality, and functionality criteria are met.
Customer	Works closely with the project manager to define requirements and desired features to be implemented.
Quality assurance testing analyst	Ensures that each release meets required standards.
Software developers	Responsible for coding and software de-bugging as well as producing user-documentation where required.
Quality control	Tests the product in line with the standards and product specifications.

Table 1. Project members' roles

During the project development life cycle, essential and important project procedures were formally documented. The product council met to address major issues as they arose, approve key phases and make strategic decisions (see Figure 1). The project involves 13 engineers, and its estimated duration is 15 months. The customer (Department of Defense) requires that minor releases be delivered within 4 to 6 weeks and that major releases not exceed six months to ensure continued funding. To meet these requirements, the product council is required to identify personnel and other resources required for the project management plan and timeline. Requirements also are prioritized and validated in line with the customer's view. Furthermore, the lowest defect density and code quality is one of the most important criteria in developing a project. As a consequence, a peer code review is required, and coding standard is enforced. The user requirements also include quick design-to- implementation cycles to ensure quick development of the product. Tight control and feedback are necessary to prevent quick development from creating quality problems. The project manager and the customer are expected to be involved in releases planning and review meetings.

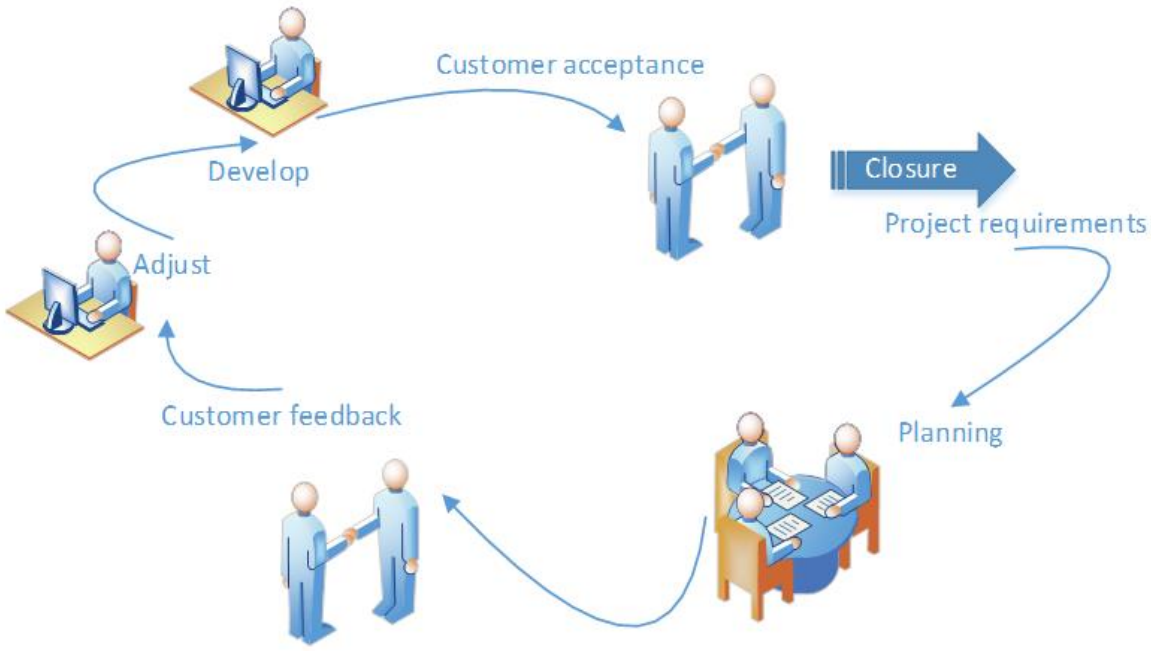

Figure 1. Project development process

Implementing a new software methodology is a process of change, including: a change in tools employed, practices followed, team workspace, and interaction and collaboration philosophy. The changes are considered a major shift in management philosophy. Given the past challenges faced by the company, the chief technology officer has *requested you to identify the most appropriate development methodology to adopt* to ensure success for the current project and achieve the following objectives:

1. **Effectiveness**: The selected system development methodology should focus on continuous customer input and improving the assessment of customer requirements better aligned to customer needs, validating and prioritizing requirements based on customer view, and delivering high value with high usability to the customer.

2. **Higher release quality**: The selected system development methodology should employ cross-functional teams where everyone on a team is responsible for building the best increment possible. Various aspects of the increment should be examined from all angles as it is developed, increasing the overall quality. Good design principle is crucial. The design should be simple, and the code should be clean and organized at all times.

3. **Increased revenue:** The selected system development methodology should result in higher success rate and more quality releases that justify premium pricing of services.

2.4 Agile Methods Overview

Agile methods for software development emerged in 1990. The focus of agile methods is on agility of software development, and here "agility means responding to changes quickly and efficiently" (cf., Qureshi, 2012). In general, agile methods are characterized by several attributes: 1) iterative, 2) incremental, 3) cooperative, 4) adaptive, 5) involving users to establish, prioritize, and verify requirements, and 5) emergent where the processes and work structure are recognized during the project, rather than pre-determined (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003; Boehm & Turner, 2005; Lindvall , Basili, Boehm, Costa, Dangle, Shull, and Zelkowitz, 2002). Agile methodology is considered a departure from older methodologies, as it is time-driven, rather requirements-driven (Dillman, 2003). It is based on iterative development, which breaks projects into a series of versions, and incremental

development, which involves designing, implementing, and testing the product incrementally until it is finished. Agile also encourages continuous revisiting of requirements during the entire life cycle of the project (Leau et al., 2012). Changing requirements, customer feedback, and early delivery of software are key points in agile methodologies (Hasnain, 2010). Agile also emphasizes cross-functional teams and value-thinking organizations (Duka, 2013). Agile methods can result in delivery of quality software, customer satisfaction, increased cost-effectiveness, cycle time reduction, and productivity (Taya & Gupta, 2011). Quality standards can be achieved through rigorous testing of the product after each cycle, with quality lessons learned fed into the next cycle (cf., Subramanian, Klein, Jiang, & Chan, 2009). To realize these benefits, the development team, businesspeople, and customers must closely collaborate and communicate (cf., ADAP Team, 2011).

2.4.1 Agile Methodologies

Traditional methodologies to develop software follow sequential steps and processes of requirement definition, design, build, and maintain. These methodologies are known by names like waterfall, plan-driven (Boehm & Turner, 2004), documentation-driven, and heavyweight methodologies (Boehm, 2002). These methodologies include extensive planning, formal processes, comprehensive documentation, and a long-term design process (Meso & Jain, 2006). There are many differences between agile methods and traditional methods. The following points summarize some of the main differences between the two approaches (cf., Boehm & Turner, 2004, 2005; Boehm, 2002; Cockburn & Highsmith, 2001; Nerur et al., 2005):

- *The development style:* The traditional methods follow a linear; life-cycle model, whereas agile methods are based on an iterative and incremental development style.
- *Requirements*: The requirements in traditional methods are knowable early and clearly defined and documented. Boehm (1988) reported, however, that requirements often change by 25% or more during their project development experience. Due to the constant changes in business environments and technology, traditional methodologies show difficulty in responding to the dynamic changes in software requirements. On the other hand, agile methods accept changing and emerging requirements to adjust to new environments.
- *Documentation*: Heavy and comprehensive documentation is required in traditional approaches, while it is less valuable than working product in agile approaches.
- *Customer involvement*: Customers are actively involved and considered team members in agile approaches, rather than they are in traditional approaches.

A variety of agile methods is available to organizations to challenge the traditional ways of developing software (Salo & Abrahamsson, 2008). Some of the most common agile methods include: eXtreme Programming (XP), Scrum, Feature Driven Development (FDD), and Crystal (Cervone, 2011). The following sections define agile methods and the differences between agile methods with respect to a set of criteria adopted from the literature.

2.4.2 eXtreme programming (XP)

XP is one of the best known and widely used agile methodologies for software development (Qureshi, 2012). C3 Team (1998) considered XP as one of the first agile methods and consists of informal planning and collecting informal customers' requirements, developing simple designs, conducting frequent unit testing and code redesign, and delivering small and simple releases of the software product in the short term. It aims at improving the quality of the software project and its responsiveness to changing customer requirements (Qureshi, 2012).

XP has four core values that support team motivation and satisfaction. These core values include (Madi, Dahalin, & Baharom, 2011):

- *Communication*: In an XP project, communication is bi-directional and based on a system of small feedback loops among team members. The customer works closely with the developers to explain and schedule the desired features. The developers handle the technical perspective. The customers communicate their satisfaction with the product progress to the development team. In an XP project, effective communication among all team members, customers, programmers and the manager is a key to the project's success (Liu & Lu, 2012).
- *Simplicity:* It stresses simple design and coding and emphasizes meeting the current functional requirements, instead of a hypothetical design.

- *Feedback:* With enough feedback, the team can measure the system and know where they are and how far the system is from the required features. Concrete feedback also allows the customer to request a change and see these requirements or adjustments implemented within a short period.
- *Courage:* This is an important value and is promoted by the other three values. It is required at all levels. With courage, the participant plays to win. XP takes courage to say, "I've done enough coding and design for today and will let the future happen." Courage enables the participant to feel comfortable because no one works alone, and changes will be adapted when they happen (Newkirk, 2002).

A number of studies have reported on the effective implementation of XP for simple and small-scale projects (Murru, Deias, & Mugheddu, 2003; Rumpe & Schroder, 2002), and for small teams. The optimal team size should consist of seven, plus or minus two members (Miller, 1956). In XP, "size clearly matters. You probably couldn't run an XP project with a hundred programmers. Nor fifty. Nor twenty, probably. Ten is definitely doable." (Beck, 2000). This is probably because XP teams are self-organizing and cross-functional, responsible for their own success and include all the expertise necessary to do so (Shore & Warden, 2008).

Furthermore, XP is proposed to address the problems of conventional development methodologies that involve long-term development and lack feedback from the client (Fernandes & Almeida, 2010). To achieve this goal, XP combines the best practices to be considered in a software project to provide the quality base for software development processes. Here is a brief summary of the major practices of XP (Beck, 1999):

- *Planning game*: This practice defines the close relationship between programmers and the customer. The developer is responsible for implementing the system, based on customer requirements within the project's constraints of time, scope, availability of technology, and team skill.
- *Small releases*: In this practice, releases are small, quickly released from daily to monthly, and achieve most customer requirements.
- *Metaphor*: This involves designing the system based on the metaphor shared between the customer and the programmers.
- *Simple design*: Code and unites test are simple.
- *Testing*: XP uses a test-driven development approach to ensure that all units' tests are satisfactory and run correctly.
- *Refactoring*: This makes the code easier to understand by removing code redundancy.
- *Pair-programming*: XP uses two programmers who write the code simultaneously.
- *Collective ownership*: Each programmer is responsible for improving the code if possible.
- *Continuous integration*: The new code should be adjusted and integrated with the existing system.
- 40-hour week: Work 40 hours a week.
- *On-site customer*: This refers to customer involvement during the project life cycle.
- *Coding standard*: Programmers write code in the same way according to agreed standards.

XP's major practices embody and realize the four core values just mentioned. The core values are a guide for the practices employed. "Each practice in and of itself can be described in terms of their adherence to the four core values of XP. They also work together to form a whole that is much greater than the sum of the parts." (Newkirk, 2002).

2.4.3 Scrum

Scrum is the leading agile development methodology for dealing with projects that have a complex innovative scope of work (Schwalbe, 2012). It "is an enhancement of the commonly used iterative and incremental object-oriented development cycle" and involves implementing a small number of customer requirements in sprint cycles of 2 to 4 weeks (Schwaber, 1995). Scrum doesn't demand specific software development practices. It does demand certain managerial practices during its different phases to avoid chaos.

Scrum includes three main roles for project participants (Schwalbe, 2012):

- *Product owner*: This individual ensures that the team delivers value to the business, and decides what work to do and in what order based on the product backlog document.
- *ScrumMaster:* This individual ensures that the Scrum process is used as expected, resolves impediments, and ensures team productivity.

- *Scrum team or development team:* The team is responsible for delivering the desired results for each sprint.

Furthermore, as in XP, Scrum involves a set of principles that lead to a higher quality of software project. These key principles are illustrated below (Fernandes & Almeida, 2010):

- *Product backlog*: The list of requirements and desired features to be implemented.
- *Effort estimation*: This is the iterative process of estimating the efforts involved in performing an item in the product backlog.
- *Sprint*: This involves performing a new increment in a project in a period of 2 to 4 weeks.
- *Daily meeting*: Daily meetings are held to track the progress of a project.
- *Sprint planning meeting*: Each sprint begins with a sprint planning meeting. The aim is to analyze and evaluate the items from the product backlog and to prepare for the next sprint.
- *Sprint backlog*: This represents a set of tasks to be executed during a sprint.
- *Sprint review meeting*: This is a review meeting at the end of each sprint.
- *Sprint retrospective*: This is to discuss the internal issues exposed during a sprint.
- *Sprint burn down chart:* A chart shows the remaining work of the sprint.

"Scrum is aimed at providing an agile approach for managing software projects while increasing the probability of successful development of software, whereas XP focuses more on the project level activities of implementing software." Both approaches, however, recognize the central principles of agile development (Salo & Abrahamsson, 2008).

2.4.4 Feature-driven development

Feature-driven development (FDD) is a model-driven, short-iteration process. It consists of five basic processes (Palmer & Felsing, 2002):

- *Developing an overall model*: The project starts with a high-level walk-through of the scope of the system and its context. Then, detailed walk-throughs are performed for each modeling area. Walk-through models are then composed by small group which presents its results for peer review and discussion. One of the proposed models or a merge of the models is selected and becomes the model for that particular domain area. Domain area models are then merged into overall models, and the overall model shape is adjusted along the way.
- *Building a features list*: The domain is decomposed into a number of subject areas. Each subject area is then partitioned into a number of activities. The step within each activity forms the categorized feature list.
- *Planning by feature:* After completing the feature list, the next step is to produce the development plan. Class ownership is accomplished by sequencing and assigning feature sets as classes to chief programmers.
- *Designing by feature:* A design package for each feature is produced. A chief programmer selects a small group of features for development. The chief programmer and class owner work out detailed sequence diagrams for the selected features. The chief programmer then refines the overall model based on the content of the sequence diagrams. The class and method prologues are written, and a design inspection takes place.
- *Building by feature:* After a successful design inspection, a complete client-valued feature is produced. The class owners develop the actual code for their classes. The code developed is then unit-tested and code-inspected. After a successful code inspection, the completed feature is promoted to the main build.

The main focus of FDD is the design and building phases, and it does not cover all software development processes (Awad, 2005). The main purpose of FDD is to deliver frequent and tangible deliverables, along with accurate tracking of the progress of reports (Awad, 2005). In FDD, customer involvement occurs in three processes: developing an overall model, building a features list, and designing by feature. The design and build iterations occur in the last two processes. Small teams are formed in the first four processes of FDD. The evolution of technology architectures also occurs in the first four processes (Rico, 2008a). Recently, FDD has a relatively small market compared to XP and Scrum.

2.4.5 Crystal

Crystal is a family of methods developed to address the specific characteristics of the project. Alistair Cockburn, one of the founder of the agile methodology movement, lists Crystal core properties for a successful project (Cockburn, 2002):

- *Frequent delivery;*
- *Close communication;*
- *Reflective improvement;*
- *Personal safety; focus;*
- *Easy access to expert users, and*
- *Technical environment with frequent integration, automated testing, and configuration management.*

Crystal methods put heavy emphasis on the importance of communication among people involved in the project. Its methods are categorized according to the project size that they address. In this context, a particular color is assigned to each member of the Crystal family to show relative complexity: The darker the color, the heavier the methodology. Clear, Yellow, Orange, Red, Maroon, Blue, and Violet are Crystal methodologies named in the literature. Crystal Orange and clear, however, have been used in real projects (Williams, 2007). There are two absolute rules of the Crystal methods: First, the use of incremental cycles must not exceed four months; and second, reflection workshops must be held after every delivery to determine what works well and what should be changed so that the method is self-adapting (Highsmith, 2002).

2.4.6 Agile methods characteristics

The need to produce higher quality, more cost-effective, and faster software solutions is leading more and more institutions to adopt agile methodology in software development (Benefield, 2008). In 2011, a "state of agile development" survey showed that more than 80% of respondents' companies were using agile methods to some extent (Rodríguez, Markkula, Oivo, & Turula, 2012). The most common agile method was eXtreme Programming (Rico, 2008b). More recently, the two most common and widely adopted agile methods were eXtreme programming and Scrum (Cervone, 2011; Fitzgerald, Hartnett, & Conboy, 2006). In addition, XP and Scrum are the most cited agile methodologies in the literature on the subject (Fernandes & Almeida, 2010), and 14% of the respondents' companies in the same survey followed a hybrid use of XP and Scrum.

The emergence of different agile methods over the past decade is evidence that the features they espouse warrant examination. We briefly presented several common features of the key agile methods earlier. These features are: development style, project team size, team distribution, customer involvement, level of documentation, and iteration time period.

Development style: The development style on agile methods is based on an iterative and incremental development process performed in a highly collaborative manner by self-organizing teams (Moniruzzaman & Hossain, 2013). Requirements and development evolve through collaboration between teams' members that allows producing high quality solutions to meet the changing needs of customers.

Project team size: Agile methods encourage small teams and small numbers of teams for projects. With small teams, less process and planning are required to plan and coordinate team members' activities (Coram & Bohner, 2005).

Team distribution: In agile methods, distribution of teams is a complex issue when an increasing number of teams from different organizations in different sites participate in a project. Several challenges can arise, including mis-communication, difficulties in coordination, work style and a country's culture.

Customer involvement: All agile methods promote high customer involvement, encouragement, and continuous, direct communication with the customer when questions arise. Allowing the customer to actively participate in the development effort is a form of customer collaboration and empowerment.

Level of documentation: Agile methods are lightweight processes that rely on a team's tacit knowledge as opposed to documentation and place more emphasis on developing the application, rather than on documentation (Boehm & Turner, 2005). Light documents are used to exchange the views. This practice reduces the consumption of resources in terms of people and time.

Iteration time period: Agile method releases schedules can be short as two weeks or long as six months. Typically, at the end of each release, customers can evaluate the products and request changes to be made to subsequent releases.

In contrast to traditional development methods, the release length, in agile methods, is fixed, but the features are not, thus helping to focus on the customer and reduce scope creep (Coram & Bohner, 2005).

Table 2 illustrates characteristics of different agile methodologies (Coffin & Lane, 2006; IIBA, 2011; Martina, 2011; Qumer & Henderson-Sellers, 2007).

Characteristic	**XP**	**Scrum**	**FDD**	**Crystal**
Development style	Iterative increments	Iterative increments	Iterative increments	Iterative increments
Project team size	Fewer than twenty people (small team)	All size	Large team	All size
Team distribution	Co-located	Co-located and Distributed team	Co-located and Distributed team	Co-located
Customer involvement	Involved	Customer involvement through the role of product owner	Customer involved through reports	Customer involved though releases
Level of documentation	Basic and as little as possible	Basic and anything of value	Vital	Basic documentation
Iteration time period	One to six weeks	Two to four weeks	Two days to two weeks	Depending on Crystal method

Table 2. Characteristics of different agile methodologies

Given the varieties of agile methodologies, choosing the best fit is a problem due to the varying needs and requirements of an organization or project. Each organization has different criteria and objectives relevant to system-development methodologies and a different order of priority for them. Rahimian and Ramsin (2008) state five factors that can have an impact on agility: team size, developers' competences, operating culture, requirements stability, and criticality of the software. Speaking practically, a software development methodology works well when it is applied to a specific situation with specific traits (Boehm, 2002).

3. A2Z Computer Equipment Company Case Teaching Notes

The learning objectives of this teaching case are to:

- Learn agile methodology and its variations. Be able to analyze business requirements and identify the most appropriate development methodology based on an analysis of the organizational setting and prioritization of multiple business objectives.
- Be able to use a multi-criteria decision tool, such as the Analytical Hierarchy Process (AHP), to systematically frame the methodology selection problem, and prioritize and evaluate system-development methodology selection objectives and alternatives.

3.1 Target Audience and Prerequisite Knowledge

The target audiences are graduate-level MIS or MBA students with a concentration in information systems. This case is appropriate in a project management or system analysis course. The case could also be used for advanced undergraduate students with a specialization in Information Systems. Students should have a system analysis and IT project management background.

Students should familiarize themselves with system development methodologies and techniques, including Agile Methodologies prior to the case discussion. Additional reading material that can be optionally provided includes Agile software development: a survey of early adopters by Vijayasarathy and Turk, 2008. In order to frame the problem as a multi-criteria problem, techniques, such as the AHP can be demonstrated before the assignment is given. The assignment should be divided into two successive class sessions, with each session about 90 minutes. In the first 30 minutes of the first session, students will form groups and discuss different Agile Methodologies and the pros and

cons of each methodology. In the next hour, student teams should be able to determine the decision criteria from the case study and construct a multi-criteria decision problem. For example, as an AHP criteria hierarchy, one valuable resource for students is Exercises for Teaching the Analytic Hierarchy Process (Bodin & Gass, 2004). The decision criteria should be weighted and prioritized based on the decision-makers' perspective. These ratings are entered into the selected decision model to produce a solution. In the first 30 minutes of the next session, it is expected that the student teams write a two-page report that presents:

- A critical analysis of the case study.
- A summary about how the student team set up the problem.
- The criteria identified, weightings for the criteria, and justification for the weights.
- Key findings, recommendation and justification.

The last hour is dedicated to group presentations and class discussion.

3.2 Sample solution

In this section, we present some sample analysis that can be used to guide post-submission discussion as well as the evaluation and assessment of student work. Specifically, we present a list of key criteria that can be inferred based on the case description and reading material for analyzing the case (see Table 3).

Criteria	**Description**
Customer involvement	The customer or product owner works closely with the project manager to define requirements and desired features to be implemented. Typically, all agile methods boost customer involvement, but with some variation. In this case, Scrum is more appropriate since customer involvement is through the role of product owner.
Project team size	The project involves small-team size, about 13 engineers. While the XP method supports small-project team size, Scrum and Crystal accept small- and large-project team size.
Product**:** • High quality releases • Simple design	High quality releases: The case sheds light on the product's quality, and the whole product should involve lowest defect density. Basically, pair programming practice in XP method aims to achieve significant advantages in increasing code-quality level and minimizing the defect density. Simple design: Team members should pay attention to continuous simple design. Typically, simple design is a XP practice.
Guarantee of success	As mentioned in the given case, the company faces a number of challenges with its previous approach such as failure to meet product quality guidelines. Agile methods, however, are a better choice to ensure a high degree of project success.
Requirements specification	The project requirements specification would be continuously revised based on customers' desire. Undoubtedly, agile methods consider requirements changing during the project life cycle.
Implementation time	The project estimated duration is 15 months. In the broad picture, agile methods enhance project completion in a short-time period.
Iteration time period	Based on the given case, the iteration time period should not exceed six weeks. Therefore, all the methodologies are appropriate.
Level of documentation	Document project procedures were required. Scrum appears more suited to achieve this criterion.

Table 3. Criteria description

A sample criteria hierarchy that can be used for pair-wise comparison is also shown in Figure 2. The proposed decision-support model to identify the most appropriate system development methodology is a multi-criteria decision model such as the Analytical Hierarchy Process. AHP "is a theory of measurement through pair-wise comparisons

and relies on the judgments of experts to derive priority scales" (Saaty, 2008). The AHP model addresses large, dynamic, and complex real world multi-criteria decision problems (Yang & Shi, 2002). It helps the decision-maker choose the optimal or best alternatives among a set of alternatives (Bodin & Gass, 2004). The AHP model starts with setting up the problem, determining the relative weights of the comparison attributes, and finally, the aggregate weights to produce final evaluation (McCaffrey, 2005).

A typical example of how to set up the problem is in Figure 2. The corresponding criteria selection descriptions are given in Table 3, and a scale to use in making judgment to determine the relative weights of the comparison attributes is in Table 4.

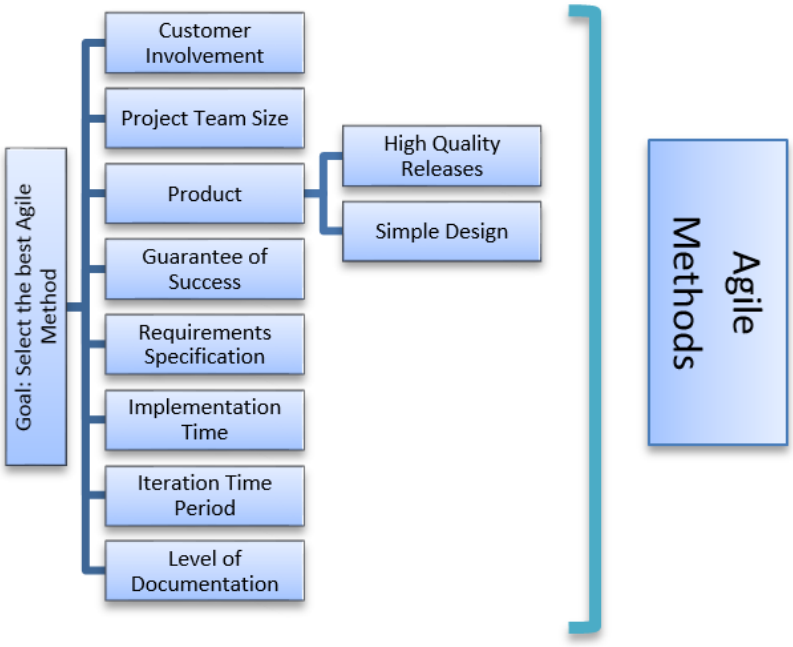

Figure 2. De-composition of the problem into hierarchy

After setting up the problem, the next step is to determine the relative weight of each attribute. AHP uses a pair-wise comparison technique. The scale to use in making judgments is shown in the following table (Saaty, 1990).

Value	Relative importance
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very importance
9	eXtreme importance
2,4,6,8	Intermediate values between the two adjacent judgments
Reciprocal	"If activity i has one of the above numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i."

Table 4. Comparison values

Since judgments about the relative weight of attributes may depend on the different agile alternatives being considered, it is important to make a judgment from the bottom up (Alanbay, 2005), which starts specifying the relative importance of alternatives with respect to the attributes, then for the attributes with respect to the goal. Only one example per level is demonstrated below.

Customer Involvement	**XP**	**Scrum**	**FDD**	**Crystal**
XP	1	1/2	1	1
Scrum	2	1	2	2
FDD	1	1/2	1	1
Crystal	1	1/2	1	1

Table 5: Pair-wise comparison of alternative with respect to the attribute "customer involvement"

The judgment value to 2 on the second row-second column means that Scrum is 2 times better in customer involvement than XP, according to the project manager. The relative weight for alternatives with respect to the sub-objectives reflects the comparison between different agile methods as described throughout the paper. The next step is to determine the relative weight of each attribute with respect to the goal.

Select best Agile Method	Customer involve-ment	Project team size	Final Product	Guarantee of success	Require-ments specifica-tion	Implemen-tation time	Iteration time period	Level of document-tation
Customer involve-ment	1	2	1	1	1	2	2	3
Project team size	1/2	1	1/2	1/3	1/2	1/2	1/2	2
Final Product	1	2	1	1	1	2	2	4
Guarantee of success	1	3	1	1	1	2	2	4
Require-ments specifica-tion	1	2	1	1	1	2	2	4
Implemen-tation time	1/2	2	1/2	1/2	1/2	1	1	4
Iteration time period	1/2	2	1/2	1/2	1/2	1	1	4
Level of document-tation	1/3	1/2	1/4	1/4	1/4	1/4	1/4	1

Table 6: pair-wise comparison of attributes with respect to choosing the best Agile Method objective

The values entered in Table 6 reflect the project manager preferences based on project requirements and case description. For example, we assumed that customer involvement is two times as important as project team size in this problem.

Expert Choice, SuperDecision, or any AHP software can be used to solve the "Best Agile Method Selection" problem. All of the above data in Tables 5 and 6 are entered to the software to calculate the weights for attributes. Figure 3 shows the solution.

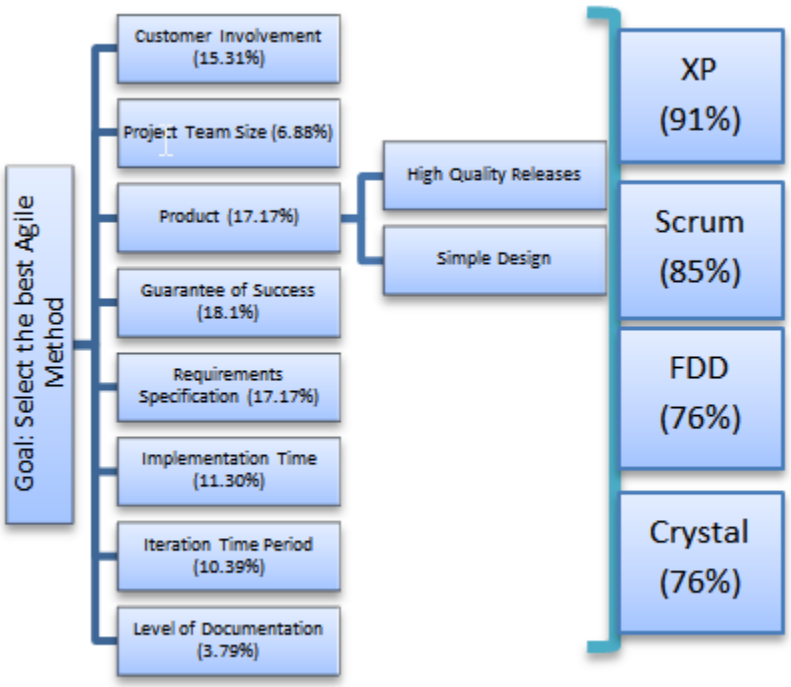

Figure 3. The solution

From the figure above, the most important criterion is Guarantee of Success. It accounts for 18.1% followed by Final Product at 17.17%. The values in parentheses next to the criteria represent their weight. From the figure above, it is obvious that the weight of XP agile model is 91%. This, however, is only an example. It should not be inferred from this that XP is better than Scrum or others methods. Since the judgments are subjectively determined, it could have been another solution.

Based on the above results, the most appropriate agile methodology, according to the listed criteria and relative judgments, is the XP model.

3.3 Evaluation criteria

The following table includes suggested grading criteria for the case analysis assignment:

Element/Grade	Grade (A)	Grade (B)	Grade (C)	Grade (D)
Decision criteria	Identifying all decision criteria from the case study.	Identifying most of decision criteria from the case study.	Identifying some of decision criteria from the case study.	Unsatisfactory analyses of case study with respect to decision criteria.
Project requirements	Prioritized list of business requirements with meaningful justification.	Prioritized list of business requirements with some justification.	Prioritized list of some business requirements.	Unsatisfactory prioritization of business requirements.

Formulating the problem	Setting up the problem using multi-criteria decision problem, such as AHP.	Achieving most of the required steps to construct the multi-criteria decision problem.	Ill-organized forming of multi-criteria decision problem.	Unsatisfactory forming of multi-criteria decision problem.
Final report	Two pages report with the required aforementioned report elements.	Two pages report with most of the required elements.	The report doesn't include the required elements.	Unsatisfactory report.
Class presentation	Class presentation and discussion.	Achieving some the required skills and presentation and discussion.	Lack of skill presentation and class discussion.	Unsatisfactory presentation and class discussion.

Table 7. Evaluation criteria

4. Conclusion

In this article, we have provided a teaching case, and we have presented a scenario for choosing the most appropriate agile method. Following the details for the case, we also present an overview so students can learn a multi-criteria decision approach to systematically frame the methodology-selection problem, and prioritize and evaluate agile selection objectives and alternatives. The case is sufficiently flexible to be used for a class assignment. Overall, the case and in-class assignment provides practice for a critical skill necessary for IS students.

5. References

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. In *Proceedings of 25th International Conference on Software Engineering*, 244–254. IEEE.

Adler, P. S., & Shenhar, A. (1990). Adapting your technological base: The organizational challenge. *Sloan Management Review, 32*(1), 25–37.

Agile Defense Adoption Proponents Team. (2011). The Business Case for Agile Methods, www.afei.org.

Alanbay, O. (2005). Selection using expert choice software. In *Proceedings of ISAHP, Honolulu, Hawaii*.

Anderson, A., Beattie, R., Beck, K., Bryant, D., DeArment, M., & Fowler, M., et al. (1998). Chrysler goes to extreme. *Distributed Computing Magazine*, *1*(10), 24–28.

Awad, M. A. (2005). *A comparison between Agile and traditional software development methodologies*. This report is submitted as partial fulfillment of the requirements for the Honours Programme of the School of Computer Science and software Engineering, The University of Western Australia.

Beck, K. (1999). Embracing change with Extreme Programming. *IEEE Computer*, *32*(10), 70–77.

Beck, K. (2000). *Extreme programming explained: embrace change*. Massachusetts: Addison-Wesley Professional.

Benefield, G. (2008). Rolling Out Agile in a Large Enterprise. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, 461–461. IEEE. doi:10.1109/HICSS.2008.382

Bodin, L., & Gass, S. I. (2004). Exercises for teaching the analytic hierarchy process. *INFORMS Transactions on Education*, *4*(2), 1–13. doi:10.1287/ited.4.2.1

Boehm, B. (2002). Get ready for agile methods, with care. *IEEE Computer*, *35*(1), 64–69.

Boehm, B., & Turner, R. (2004). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In *Proceedings of 26th International Conference In Software Engineering*, 718–719. IEEE.

Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *Journal of Software*, *22*(5), 30–39.

Boehm, B. W. (1988). Understanding and controlling software costs. *Journal of Parametrics*, *8*(1), 32–68.

C3Team. (1998). Chrysler goes to extreme. *Distributed Computing Magazine*, *1*(10), 24–28.

Cervone, H. F. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services*, *27*(1), 18–22. doi:10.1108/10650751111106528

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*(6), 961–971.

Cockburn, A. (2002). *Agile software development*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *IEEE Computer*, *34*(11), 131–133.

Coffin, R., & Lane, D. (2006). A practical guide to seven Agile methodologies. Retrieved from http://www.devx.com/architect/Article/32836

Coram, M., & Bohner, S. (2005). The impact of agile methods on software project management. In *Proceedings of 12th IEEE International Conference and Workshops In Engineering of Computer-Based Systems*, 363–370. IEEE.

Dillman, A. E. (2003). Selecting a Software Development Life Cycle (SDLC) methodology a Practical decision framework to maximize business value. PM Solution Technology White Paper Series.

Duka, D. (2013). Adoption of agile methodology in software development. In *Proceedings of 34th International Convention on Information and Communication technology, electraonics and microelectronics*. IEEE.

Fernandes, J. M., & Almeida, M. (2010). Classification and comparison of Agile methods. In *Proceedings of 2010 Seventh International Conference on the Quality of Information and Communications Technology*, 391–396. IEEE. doi:10.1109/QUATIC.2010.71

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 200–213. doi:10.1057/palgrave.ejis.3000605

Hajjdiab, H., & Taleb, A. S. (2011). Agile adoption experience: A case study in the U.A.E. In *Proceedings of 2011 IEEE 2nd International Conference on Software Engineering and Service Science*, 31–34. IEEE. doi:10.1109/ICSESS.2011.5982247

Hasnain, E. (2010). An overview of published agile studies: A systematic literature review. In *Proceedings of the 2010 National Software Engineering Conference*, 10. ACM.

Highsmith, J. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Holodnik-Janczura, G., & Golinska, I. (2010). Decision support system for choosing a model for a software development life cycle. *Operation Research and Decisions*, (1), 61–77.

International Institute of Business Analysis (IIBA). (2011). Agile Extension to the BABOK guide. Retrieved from www.iiba.org

Leau, Y., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle agile vs traditional approaches. In *proceddings of International Conference on Information and Network Technology*, 162–167. Singapore.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., & Zelkowitz, M. (2002). Empirical findings in agile methods. In *Proceedings of Extreme Programming and Agile Methods—XP/Agile Universe*, 197–207.

Liu, L., & Lu, Y. (2012). Application of agile method in the enterprise website backstage management system. In *Proceedings of 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, 2412–2415. IEEE.

Madi, T., Dahalin, Z., & Baharom, F. (2011). Content analysis on agile values: A perception from software practitioners. In *Proceedings of2011 Malaysian Conference in Software Engineering*, 423–428. IEEE. doi:10.1109/MySEC.2011.6140710

Martina, S. (2011). *Agile methodology*. University of Zagreb. Retrieved from http://agile-only.com/master-thesis

McCaffrey, J. (2005). The Analytic Hierarchy Process. *Msdn Magazine*.

Meso, P., & Jain, R. (2006). Agile software development: Adaptive systems principles and best practices. *Information Systems Management*, *23*(3), 19–30.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81.

Moniruzzaman, A. B. M., & Hossain, D. S. A. (2013). Comparative Study on Agile software development methodologies. *arXiv Preprint arXiv:1307.3356*.

Murru, O., Deias, R., & Mugheddu, G. (2003). Assessing XP at a European internet company. *IEEE Softw*, *20*(3), 37–43.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78.

Newkirk, J. (2002). Introduction to agile processes and extreme programming. In *Proceedings of the 24th international conference on Software engineering*, 695–696. ACM.

Palmer, S., & Felsing, J. (2002). *A practical guide to feature driven development*. Prentice Hall.

Project Management Institute (PMI). (2014). PMI Agile Project Management Toolbox. Retrieved from http://www.pmi.org/Certification/New-PMI-Agile-Certification/PMI-Agile-Toolbox.aspx

Purcell, J. (2012). Comparision of software development lifecycle methodologies. SANS Institute.

Qumer, A., & Henderson-Sellers, B. (2007). Measuring agility and adoptability of agile methods : A 4-dimensional analytical tool. In *Proceedings of IADIS International Conference Applied Computing*, 503–507.

Qureshi, M. R. J. (2012). Agile software development methodology for medium and large projects. *IET Software*, *6*(4), 358–363.

Rahimian, V., & Ramsin, R. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. In *processdings of Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference*, 337–342. IEEE.

Rico, D. (2008a). Effects of agile methods on website quality for electronic commerce. In *proceedings of the 41st Annual Hawaii International Conference on System Sciences*.

Rico, D. (2008b). What is the ROI of agile vs. traditional methods. *TickIT International*, *10*, 9–18.

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in finnish software industry. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, 139–148. ACM.

Rumpe, B., & Schroder, A. (2002). Quantitative survey on extreme programming projects. In *Proceedings of Third Int. Conf. on Extreme Programming and Flexible Processes in Software Engineering (XP2002)*, 95–100. Alghero, Italy.

Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, *48*(1), 9–26. doi:10.1016/0377-2217(90)90057-I

Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, *1*(1), 83. doi:10.1504/IJSSCI.2008.017590

Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *Software, IET*, *2*(1), 58–64.

Schwaber, K. (1995). SCRUM development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*.

Schwalbe, K. (2012). Managing a project using an agile approach and the PMBOK® Guide. In Proceedings of the Information Systems Educators Conference ISSN. 1435.

Shore, J., & Warden, S. (2008). *The art of Agile development*. O'Reilly.

Subramanian, G. H., Klein, G., Jiang, J. J., & Chan, C. L. (2009). Balancing four factors in system development projects. *Communications of the ACM*, *52*(10), 118–121.

Taya, S., & Gupta, S. (2011). Comparative Analysis of Software Development Life Cycle Models. *International Journal of Computer Science and Technology*, *2*(4), 536–539.

Vijayasarathy, L., & Turk, D. (2008). Agile software development: A survey of early adopters. *Journal of Information Technology Management*, *XIX*(2), 1–8.

Wei, H., Xin, C., & Ying, H. (2010). Non-computer professional IT education in the MBA model. In *Proceedings of 2010 5th International Conference on Computer Science & Education*, 612–614. IEEE. doi:10.1109/ICCSE.2010.5593537

Williams, L. (2007). A survey of agile development methodologies. Retrieved from http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf

Yang, J., & Shi, P. (2002). Applying analytic hierarchy process in firm's overall performance evaluation : A case study in China. *International Journal of Business*, *7*(1), 29–46.

Page intentionally left blank