# Assessing Agile Methods: Investigating Adequacy, Capability, and Effectiveness

***An Objectives, Principles, Strategies Approach***

Shvetha Soundararajan

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

May 7, 2013
Blacksburg, VA

Keywords: Agile Assessment; Adequacy; Capability; Effectiveness; Objectives; Principles; Strategies; Indicators

# Assessing Agile Methods: Investigating Adequacy, Capability, and Effectiveness

***An Objectives, Principles, Strategies Approach***

Shvetha Soundararajan

ABSTRACT

Agile methods provide an organization or a team with the flexibility to adopt a selected subset of principles and practices based on their culture, their values, and the types of systems that they develop. More specifically, every organization or team implements a customized agile method, tailored to better accommodate its needs. However, the extent to which a customized method supports the organizational objectives, i.e. the 'goodness' of that method, should be demonstrable. Existing agile assessment approaches focus on comparative analyses, or are limited in scope and application. In this research, we propose a systematic, comprehensive approach to assessing the 'goodness' of agile methods. We examine an agile method based on (1) its adequacy, (2) the capability of the organization to support the adopted principles and strategies specified by the method, and (3) the method's effectiveness. We propose the Objectives, Principles and Strategies (OPS) Framework to guide our assessment process. The Framework identifies (a) objectives of the agile philosophy, (b) principles that support the objectives and (c) strategies that implement the principles. It also defines (d) linkages that relate objectives to principles, and principles to strategies, and finally, (e) indicators for assessing the extent to which an organization supports the implementation and effectiveness of those strategies. The propagation of indicator values along the linkages provides a multi-level assessment view of the agile method. In this dissertation, we present our assessment methodology, guiding Framework, validation approach, results and findings, and future directions.

*"To measure is to know."*

*"If you can not measure it, you can not improve it."*

*- Lord Kelvin*

*To Appa*

# Acknowledgements

Dr. Arthur, my advisor is an exceptional guru. I appreciate his patience in guiding me in my research and writing. He has inspired me to think independently and critically, to have an open mind, and to ask questions and voice my opinions. Working with him has been a privilege.

My committee members have been instrumental in shaping this research. I consider Dr. Balci my second advisor. I thank him for his advice on a broad range of topics from assessment methodologies to purchasing the right macbook. I am grateful to Dr. Tilevich for offering research and career guidance, Dr. Stevens for his valuable suggestions, and Dr. Sheetz for his insights from a business perspective.  I also thank them all for a highly memorable final defense session.

The Department of Computer Science at Virginia Tech has been my home away from home. I have had a wonderful time here as a graduate student. I am grateful for the opportunity to contribute to the teaching mission of the department as a Teaching Assistant. I have enjoyed every minute of it. Drs. Ribbens and Ramakrishnan have both been very supportive and encouraging. I thank Sharon Kinder-Potter, Melanie Darden, Julie King, Emily Simpson, Jesse Eaves, Terry Arthur, and Libby Bradford for all their help with paperwork, travel reimbursements, and logistics.

My colleagues and friends are my support system here. Dr. Amine Chigani has been a great friend, philosopher, and mentor. His guidance over the years has been invaluable and I will always be grateful for that. I thank Jeremy Barksdale and Serdar Aslan for being sounding boards for my research ideas. Thanks to Tram Bethea and Vinh To who have helped substantiate the linkages presented in this research.

Aparna Mahadevan and Sreeram Ramalingam are among the very few people who have supported my decision to pursue my doctoral degree. I can never forget their help and I am thankful for the same. I am fortunate to have wonderful friends like Pavithra Sekhar, Kalyani Nagaraj, and Shubhangi Deshpande who have been very supportive. We have had an eventful time here at Virginia Tech and I will always cherish these memories. I am grateful to my friend Tilottama Gaat for her help in identifying a suitable validation site for my research.

Words cannot express my love and gratitude for my parents and my sister. Saying “Thank You” seems inadequate and paltry. My mother is my lifeline. Her love and her confidence in me have guided me through times of happiness and disappointments. My father has always encouraged me to aim high, and to be a leader and not a follower. I am blessed to have his unconditional love and support. Though he did not live to see me graduate, I know that he is always with me. I am blessed to have the most wonderful younger sister. Her support has been invaluable. Hearing her voice always brightens my day. I thank God for giving me such a wonderful family. The credit for all that I have achieved so far goes to them.

I also thank my extended family for their help and support over the years.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

The formulation of the agile manifesto [1] has given rise to a plethora of software development methodologies that propose to embody the values and principles of the agile philosophy. Practitioners have a wide range of agile methods and practices to choose from. However, there is no apparent consensus on the values, principles, and practices an individual, an organization or a team should adopt because each agile method is most often tailored for specific domains and development environments. That is, there is no "one-size-fits-all" approach to agile software development. While this flexibility is consistent with the agile philosophy, it can lead to the adoption of principles and strategies that can be sub-optimal relative to the desired objectives. Researchers have observed that numerous organizations and teams adopt and use agile practices without a proper understanding of the implications of their use [2, 3]. The absence of a single agile method has resulted in organizations having a divergent view of "being agile." Many organizations proclaim themselves to be 'agile' when in fact they are following what is considered "Agilefall", that is, using agile terms but essentially following the waterfall model [4]. Moreover, many practitioners equate "being agile" with 'no planning' and 'no documentation.' We question then, how can one determine if adopted practices are consistent with the identified principles, and to what extent those principles support organizational objectives? Existing agile assessment approaches are limited in scope and application. More specifically, they assess, in part, certain aspects of the agile software development process. Also, most approaches focus on assessing the product and potentially ignoring measures reflecting the characteristics of the people or the process. Hence, there is a need for a comprehensive agile assessment approach.

This dissertation presents a structured, systematic, and comprehensive agile assessment process that examines *people, process, project, product, and environment* characteristics within organizations adopting agile methods. Our inclusive *approach can be used by organizations to assess their adopted agile method, identify the effective components of their development process, as well as reveal possible inadequacies, and potential enhancements.* In this research, we focus on assessing the 'goodness' of an agile method adopted by an organization based on *(1) its adequacy to achieve the enunciated objectives,* *(2) the capability of the organization to provide the supporting environment to successfully implement that method,* and *(3) the method's effectiveness*.

To guide our assessment, we propose the **O**bjectives, **P**rinciples, and **S**trategies (OPS) framework. The design of the OPS framework revolves around the identification of the agile objectives, principles that support the achievement of those objectives, and strategies that are implementations of those principles. Well-defined linkages between the objectives and principles, and between the principles and strategies are also established to support the assessment process. We assess the *adequacy* of an agile method by traversing the linkages in a top-down fashion. That is, given the set of objectives espoused by the agile method, we follow the linkages downward to ensure that the appropriate principles are enunciated, and that the proper strategies are expressed. We assess the *capability* of an organization to implement its adopted method and the *effectiveness* of that implementation by using both a top-down and bottom-up traversal of the linkages. The bottom-up assessment, however, is predicated on the identification of people, process, project, product, and environment properties associated with each strategy that attest to the presence and implementation of that strategy. We refer to each (strategy, property) pair as an indicator. By following the linkages upward from the indicators, we can infer the use of proper principles and the achievement of desired objectives.

## *1.1 Motivation*

Over the past ten years, we have seen a significant rise in the number of organizations transitioning to agility. Some of the reasons for the increasing agile adoption rates are *(1) the ability to accommodate change throughout the development lifecycle, (2) improved quality, (3) greater return on investment, (4) shorter development periods, (5) improved customer satisfaction, (6) better team morale, (7) reduced waste* and *(8) better predictability* [5-7]. However, people in many of these organizations have observed that their agile adoption efforts are not as effective as they should have been, or have often failed to yield the expected results. Agile adoption in an organization is guided by its culture, values, and application domains. The agile philosophy provides an organization or a team the flexibility to tailor an agile method to better suit its goals and objectives. More often than not, however, the appropriate agile principles, implementation strategies, and strategies are not reflected by the customized methods. In effect, these tailored methods may be sub-optimal relative to achieving desired objectives. Also, organizations often lack the supporting environment to effectively employ the adopted strategies. As a result, the benefits afforded by agile strategies are not fully realized [8]. Hence, we consider it prudent to question the extent to which an agile method or a customized approach satisfies the needs of an organization. In effect, we would question the 'goodness' of that approach.

The agile philosophy places utmost importance on "working software" as being the primary measure of progress. Hence, most assessment approaches for agile methods focus on assessing the developed *product*, and largely ignore potential measures reflecting process, project, and people characteristics. For those few approaches that do assess the *process*, the focus is to determine the presence or absence of agile strategies in an organization rather than the degree to which those strategies are effectively used in order to achieve desired objectives. Hence, there is a significant need for a more comprehensive and integrated approach to assessing the 'goodness' of agile methods.

## *1.2 Problem Statement*

An agile method is composed of a set of objectives that the method proposes to achieve, principles that support the achievement of those objectives, and strategies that are implementations of the stated principles. An organization's culture (people and their attitudes), its values (beliefs and ideas about what kind of goals should be pursued), and the types of systems being developed (small-, medium- or larger-scale, mission and life-critical, network-centric, etc.), define the composition of an adopted agile method. Given the various factors that shape an agile method within an organization, how can we determine if that method is consistent with the organizational objectives? ***In short, how can we assess the 'goodness' of an agile method?***

In assessing the 'goodness' of an agile method, we examine the method from three perspectives:

1. *We review the composition of the agile method independent of any organizational objectives.*

   The agile community has endorsed many standalone agile methods such as eXtreme Programming, Feature Driven Development, Crystal, Lean, etc. How can organizations intending to adopt any of those accepted methods, or attempting to design a customized approach, ensure that the method is sufficient with respect to meeting its stated objectives?

2. *We study an instance of the agile method defined by the organization.*

   An instance of an adopted agile method should reflect the culture and values of the organization. Hence, the objectives, principles, and strategies touted by the instance may differ from the composition of the method in its original form.

   Does the instance state desired objectives that reflect the organizational objectives, and the principles that support those objectives?

Given an instance of the agile method, does the organization have the ability to support its implementation?

3. *We observe the effect of the instantiated method within the organization.*

   Did the application of the method yield the expected/ intended results?

   Does the organization's supporting environment and/or the adopted strategies impede or support producing the expected results?

The problem then is to provide organizations with a systematic approach to assessing their adopted agile method so that they can determine if their method is consistent with their organizational objectives.

## 1.3 Issues

In defining an approach to assessing the 'goodness' of agile methods, we have identified the following issues to be addressed:

1. *Determining the assessment perspectives*

   A primary concern in this research has been to ascertain the perspectives from which the 'goodness' of an agile method can be assessed. As discussed previously, we propose to examine (1) a method independent of an organization, (2) an instance of a method within an organization, and (3) the implementation of that instantiated method. This respectively entails (1) assessing the sufficiency of the method with respect to meeting stated objectives, (2) evaluating the ability of an organization to support the implementation of the method, and (3) determining if the method produced the intended results.

2. *Recognizing objectives, principles and strategies*

   We recognize the existence of objectives embodied by agile methods, principles used to support the objectives, and strategies employed that are reflective of the principles. The agile manifesto [1] provides four focal values that form the core of the agile philosophy and a set of twelve supporting principles. Our task is to derive a set of objectives that are reflective of the agile philosophy and the focal values as stated by the manifesto. Also, from the manifesto, we have to identify a comprehensive set of principles that support the objectives.

3. *Establishing the relationships between the objectives, principles and strategies*

   This research focuses on determining if an agile method adopted by an organization is consistent with the organizational objectives. Hence, given an agile method with its set of objectives, principles, and strategies, relationships among them have to be established in order to determine if there are principles that support the objectives, and strategies that are implementations of the principles. The relationships between objectives, principles, and strategies are not explicitly stated in the existing literature. Hence, we have to definitively identify and establish the relationships and provide evidence for the same.

   We realize that for an objective and the set of principles associated with that objective, one or more principles may support that objective to a greater extent than the others. Similarly, for each principle and the set of strategies that are related to that principle, some strategies may be necessary for the achievement of that principle. Recognizing that some relationships between the objectives, principles and strategies may be more important than the others, we have to address this disparity in our solution approach.

4. *Identifying the observable characteristics of the people, process, project and product*

   The objectives, principles, and strategies are concepts that are at higher levels of abstraction. In order to assess agile methods, the mechanics of implementation of those methods and the environment within which those methods are applied have to be studied. More specifically, observable characteristics of the people, process, project, product, and the environment that are associated with the strategies have to be identified. When used, each strategy induces an associated set of observable characteristics. We need to identify these characteristics so that we can assess:

   - An organization's ability to support the implementation of an agile method.
     - This depends on the people, process, project, and environment characteristics.

   - The extent to which the method produces the expected results.
     - This depends on the process artifacts and product characteristics

5. *Validating the assessment approach*

   Validating the assessment approach primarily involves (1) substantiating the assessment approach, and (2) gathering evidence for the existence of the objectives, principles, and strategies, the relationships among them, and the observable characteristics. To determine the ability of an organization to support an instantiation of an agile method and evaluate the extent to which that instance produces the expected results, we have to study the method and its implementation within an organization. Hence, to validate the above-mentioned assessment perspectives, we require the application of the assessment approach in an industrial setting.

The solution approach described in the next section outlines our proposed approach to address the above-mentioned issues.

## *1.4 Solution Approach*

We advocate the need for a more comprehensive agile assessment process that assesses the *people, process, project, product, and environment* characteristics of organizations adopting agile methods. In this research, we describe an approach for assessing the 'goodness' of agile methods from three perspectives. More specifically, to assess the collective 'goodness' of a given agile method, we address the following three questions:

- How ***adequate*** is the method with respect to achieving its objectives?
- How ***capable*** is an organization in providing the supporting environment to implement that method?
- How ***effective*** is the implementation of the method in achieving its objectives?

In response to the above questions, we present the **O**bjectives, **P**rinciples, and **S**trategies (**OPS**) Framework to guide the assessment of adequacy, capability, and effectiveness of agile methods. The OPS Framework is hierarchical and identifies (a) at the first level, objectives of the agile philosophy, (b) principles that govern the achievement of those objectives at the second level, and (c) at the third level, strategies that implement those principles. The core structure of the OPS Framework is illustrated in Figure 1.1. The Framework includes linkages that signify definitive relationships between identified objectives and principles, and between principles and strategies. In Figure 1.1, the arrows between the objectives, principles, and strategies indicate the existence of linkages. These linkages are central to our assessment process. For each strategy, we also define indicators to assess the extent to which an organization supports its implementation and the effectiveness of that strategy.

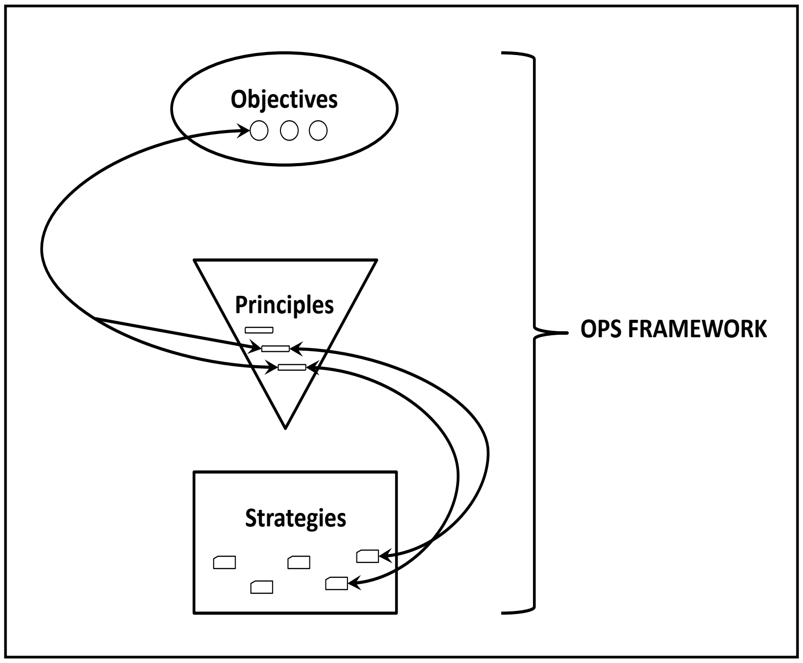


***Figure 1.1. Core structure of the OPS Framework***

We assess the *adequacy* of an agile method by traversing the linkages in a top-down fashion. That is, given the set of objectives enunciated by the agile method, we follow the linkages downward to ensure that the appropriate principles are identified, and that the proper strategies are expressed. In addition to a top-down examination, the *capability* of an organization to implement its adopted method, and the method's *effectiveness,* are assessed using a complementary bottom-up traversal of the linkages. This begins, however, by identifying people, process, project, and product properties that attest to the use of particular strategies. Then, by following the linkages upward from the strategies, we can infer the use of proper principles and the achievement of desired objectives.

To better understand the applicability of the OPS Framework, we identify two complementary but distinctive categories of agile methods:

- *Basis Methods* – These are standalone methods that have been endorsed by the agile community, e.g. eXtreme Programming (XP) [9], Lean [10], Crystal [11], and Feature Driven Development (FDD) [12].
- *Instance Methods* – These are instantiations of an agile method within an organization. As mentioned previously, organizations and teams customize an agile method to suit their needs. Such methods are usually modified instances of one or more Basis methods and differ with respect to their objectives, principles, and strategies.

Using the OPS Framework, we can assess the adequacy of both Basis and Instance methods. We assess adequacy by examining *the agile method under consideration to ensure that the method advocates the appropriate principles and strategies necessary to achieve its touted objectives.* Hence, organizations or teams can examine the composition of any agile method and confirm the presence (or absence) of the necessary principles and strategies that support the achievement of that method's stated objectives. Unlike adequacy, capability and effectiveness are assessed from an organizational perspective. That is, we can assess the capability and effectiveness only of the Instance methods. To assess capability, we examine *an organization's or a team's resources and competencies to determine the ability of that organization or team to support the implementation of its adopted agile method*. The effectiveness of an agile method is determined by *the extent to which the implementation of that method by an organization or team produces the expected results*.

Currently, our substantiation of the OPS Framework and the assessment process has been a three-pronged process:

- *Substantiating the components of the OPS Framework*

  An online survey was administered to gather the feedback from agile practitioners about the core components of the OPS Framework. The results provided in Chapter 5 show that the identified objectives, principles, and strategies are consistent with the practitioners' views of the agile philosophy and development paradigm.

  To establish the existence of the identified linkages between the objectives and the principles, and the principles and the strategies, we have used sources including, but not limited to, the agile manifesto, books, research papers, experience reports, white papers and discussions with industry experts. The complete set of established linkages with the evidence for their existence can be found in Appendix B.

  Additionally, to gather practitioners' views regarding the viability and utility of the OPS Framework and assessment methodology, we administered two other surveys – one at the Agile conference held at Dallas, TX in 2012, and the other at Company Z. Most practitioners who completed the surveys agree that (a) there is a need for a comprehensive agile assessment approach and (b) the application of the assessment methodology developed in this research is viable in an industry setting. See Chapter 5 for the results.

- *Applying the Adequacy assessment process*

  We have assessed the adequacy of three Agile methods – eXtreme Programming (XP), Feature Driven Development (FDD), and Method A which is a tailored instance of XP. The former two are Basis methods and the latter is an Instance method. Applying our adequacy assessment process, we have determined that (a) XP is better at supporting its objectives than either FDD or Method A, and that (b) Method A is more effective (or adequate) in supporting its objectives than FDD (see Chapter 6).

- *Substantiating the assessment methodology*

  We have designed and implemented an onsite examination and assessment of agile methods adopted by Company Z in Blacksburg, VA. Based on our observations recorded during the three-month onsite study at Company Z, we have assessed the adequacy, capability, and the effectiveness of agile methods adopted by *four* agile teams. Results (see Chapter 7) show that the OPS based assessments are consistent with the perceptions of Company Z employees, and the Subject Matter Experts (SMEs). This establishes the validity of the OPS Framework and assessment methodology.

Results from our substantiation of the *components of the OPS Framework* and the *adequacy assessment* of XP, FDD and Method A can be found in Chapter 6 and also in [13].

## 1.5 Blueprint

The remainder of this dissertation is organized as follows:

- In Chapter 2, we review some of the existing agile assessment approaches and the two approaches guiding the OPS Framework.
- Chapter 3 provides an overview of the OPS framework and its components. It also outlines our approach to assessing ‘goodness’.
- In Chapter 4, we present the results from substantiating the components of the OPS Framework.
- Chapter 5 describes the application of the adequacy assessment process to three agile methods.
- Results from our onsite study are detailed in Chapter 6.
- In Chapter 7, we discuss the main contributions of our work, future directions, and finally summarize our work.

## References

[1] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn*, et al.* (2001), *Manifesto for Agile Software Development*, http://www.agilemanifesto.org

[2] S. Soundararajan and J. D. Arthur (2011), "A Structured Framework for Assessing the "Goodness" of Agile Methods," in *18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, pp. 14-23.

[3] A. Sidky, J. D. Arthur, and S. Bohner (2007), "A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework.," *Innovations in Systems and Software Engineering,* vol. 3, pp. 203-216.

[4] J. Little (2007), *The Nokia Test,* http://agileconsortium.blogspot.com/2007/12/nokia-test.html

[5] S. Soundararajan and J. D. Arthur (2009), "A Soft-Structured Agile Framework for Larger Scale Systems Development," in *16th Annual IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS)*, San Francisco, pp. 187-195

[6] A. Sidky and J. D. Arthur (2007), "Determining the Applicability of Agile Practices to Mission and Life-Critical Systems," presented at the Proceedings of the 31st IEEE Software Engineering Workshop.

[7] C. Schwaber (2007), "The Truth About Agile Proceses - Frank Answers to Frequently Asked Questions," Forrester Research.

[8] A. Sidky and J. D. Arthur (2008), "Value-Driven Agile Adoption: Improving An Organization's Software Development Approach," presented at the New Trends in Software Methodologies, Tools and Techniques - Proceedings of the Seventh SoMeT, Sharjah, United Arab Emirates.

[9] K. Beck and C. Andres (2004), *Extreme Programming Explained: Embrace Change*, 2 ed.: Addison-Wesley Professional.

[10] M. Poppendieck and T. Poppendieck (2003), *Lean Software Development: An Agile Toolkit*: Addison Wesley.

[11] A. Cockburn (2004), *Crystal Clear: A Human-Powered Methodology for Small Teams*: Addison-Wesley Professional.

[12] S. R. Palmer and J. M. Felsing (2002), *A Practical Guide to Feature Driven Development*. Upper Saddle River, New Jersey: Prentice Hall PTR.

[13] S. Soundararajan, J. D. Arthur, and O. Balci (2012), "A Methodology for Assessing Agile Software Development Methods," in *Agile Conference (AGILE) 2012*, pp. 51 - 54.

# 2 Background

The need for a structured, systematic and comprehensive approach to assessing the ‘goodness’ of agile methods forms the basis of our work. In this chapter, we review some of the current agile assessment approaches that have motivated our research. Additionally, to help readers better understand the work presented in this document, we provide background information about the agile philosophy, its values and principles. We also present an overview of the Objectives Principles and Attributes Framework and the Evaluation Environment methodology that guide the construction and design of the OPS Framework and the assessment approach (Chapter 3).

## *2.1 Agile Overview*

“Software engineering is (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and (2) the study of approaches in (1)” [1]. The Software Development Lifecycle (SDLC) was introduced in order to structure the software development process and ensure that the process complies with the above definition. The software development process systematically begins with identifying and specifying user requirements and then proceeds to the architecting, design, coding, testing and maintenance phases. Traditional software development methodologies such as the Waterfall model [2] and the Spiral model [3] have helped reduce the chaotic state of software development by focusing on:

- Extensive planning to outline the process to be followed, identifying tasks to be completed during product development, and determining product milestones,
- Comprehensive documentation in the form of Software Requirements Specification (SRS), high and low-level design documents, test plans, etc.,
- Gathering and specifying user requirements upfront before proceeding to the architecting phase,
- Anticipating future requirements and designing the architecture of the system to accommodate current and future requirements, and
- Process monitoring and control, risk management, and software quality control and assurance.

However, in the last few years, many development teams have realized that these traditional approaches to software development are not suitable to all [4]. The traditional approaches are found to be inadequate and expensive for development efforts that have to address rapidly changing requirements. Conventional methods attempt to foresee requirements and create the software system architecture upfront in order to accommodate current and future requirements. However, when previously unidentified requirements surface, the current architecture may no longer be valid. The cost of modifying the architecture, and in turn the code, in order to accommodate requirements identified late during the development lifecycle is very high [3].

Creating comprehensive documentation is often a wasted activity in the face of changing requirements. Documents have to be maintained regularly to reflect any changes made to requirements and design. In effect, maintaining comprehensive documentation is expensive. Also, organizations and teams involved in developing small-scale systems find the extensive planning and documentation efforts cumbersome.

The above-mentioned issues prompted the need for new cost-effective, lightweight methods to accommodate rapidly changing requirements in software systems. This realization motivated the development of the "**Agile**" approach, which is best described by the "Manifesto for Agile Software Development" [5] as given below:

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

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

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

The core values of the agile manifesto are briefly discussed below:

1. The agile movement focuses on fostering team spirit among the members of the software development team by stressing close relationships and open working environments. Human aspects of the software development process are considered more important than the process itself.

2. The main objective of the software development team is to produce working software at regular intervals. Agile methods advocate the adoption and use of iterative and incremental development approaches. Working software is used to measure progress and minimal documentation is produced.

3. Relationships between the customers and the development team are given preference over strict contracts. Agile methods accentuate delivering software that would provide maximum business value to the customers thereby reducing the risk of not fulfilling the contract. Also, the customer is involved throughout the development process, which fosters better customer-developer relationships.

4. The customers and the development team should be prepared to modify plans in order to accommodate change even late in the development lifecycle.

Agile methods are lightweight processes that focus on *short iterative cycles, direct customer and user involvement, incremental development and accommodating change even late in the development lifecycle*. Also, these methods are value-driven; that is, they focus on maximizing the business value to the stakeholders involved. “Business value is something that delivers profit to the organization paying for the software in the form of an Increase in Revenue, an Avoidance of Costs, or an Improvement in Service (IRACIS) which are discussed below:

- Increase Revenue – Will this functionality increase the revenue generated by the system?
- Avoid costs – Will this functionality improve efficiency and reduce wasteful processes?
- Improve Service – Will this application help us provide timely service and information to others in a better way?” [6].

In the field of software development, there has been a shift in focus from conventional software engineering approaches towards agility. The realization that the traditional plan-driven approaches to software engineering are not suitable to all organizations and teams has motivated the creation of the Agile Manifesto. The creation of the manifesto has given rise to many agile methods like eXtreme Programming (XP) [7-9], Scrum [9, 10], Crystal methodologies [9, 11, 12], Feature Driven Development [9, 13], Lean Development [14, 15], etc.

***A Generic Agile Process***

In the following paragraphs, we describe a generic agile process. The information presented here is based on readings from [9, 16-19]. Figure 2.1 shows a generic agile process that follows an iterative and incremental approach. At the start of a project, all the stakeholders meet to decide on its scope. The time required to develop the product is estimated and then divided into multiple release cycles. During each release cycle, a potentially shippable increment of the product is developed. A release cycle is composed of multiple iterations. An iteration is a short software development cycle that lasts two to four weeks as shown in Figure 2.1.

During the initial meeting, often referred to as "***Iteration 0***", the stakeholders identify a set of features. These features are sets of functionality that deliver business value to the customer and are stored on a "***Project Backlog"*** (Figure 2.1). These features are then prioritized and the time required for the development of each is determined. Also, the set of features to be developed during the first release cycle is established.

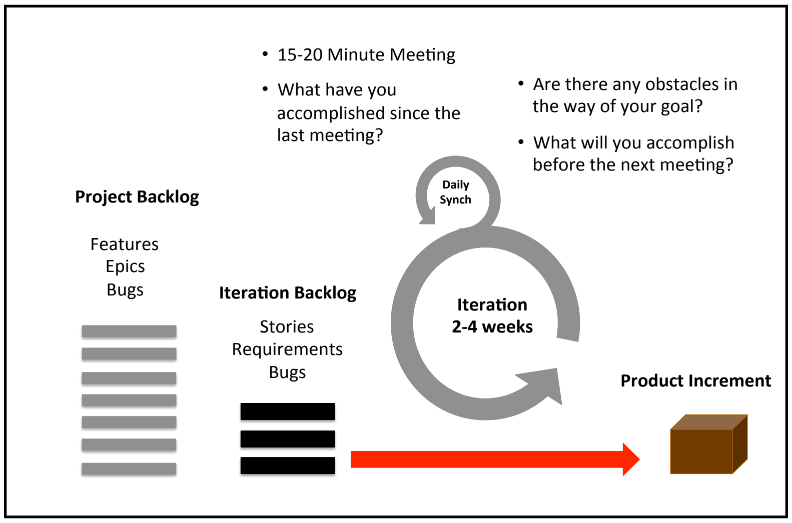

• 15-20 Minute Meeting
• What have you accomplished since the last meeting?
• Are there any obstacles in the way of your goal?
• What will you accomplish before the next meeting?
Daily Synch
Project Backlog
Features
Epics
Bugs
Iteration Backlog
Stories
Requirements
Bugs
Iteration
2-4 weeks
Product Increment

***Figure 2.1. A generic agile process***

Each feature is decomposed into multiple stories. Stories are brief descriptions of user- or customer-valued functionality. The time required to implement each story is determined and the stories are then prioritized. The prioritized stories are stored on the "Iteration Backlog" (see Figure 2.1), which is specific to each iteration. Each story may be further decomposed into tasks. Tasks are essentially checklists for developers, which outline the details required for implementing the stories. During an iteration, the stories are developed and are integrated with the existing code base on a daily basis as indicated by the "Daily Synch" loop in Figure 2.1. Also, as shown in Figure 2.1, the members of the development team meet everyday for 15 to 20 minutes to discuss:

(1) Their accomplishments since the last meeting
(2) The issues (if any) that they faced, and
(3) Their next tasks to be completed.

Agile methods focus on customer communication and collaboration. Therefore, customers or customer representatives are usually present onsite to answer any questions the development team may have and participate in the product development lifecycle. The customers are responsible for creating and recording the acceptance criteria for the stories and features. Developers write tests to meet the acceptance criteria defined by the customers. Also, at the end of each iteration, the customers test the software produced thus far against the acceptance criteria that they outlined previously.

Bugs identified when testing may be stored in the iteration backlog if they can be fixed during the same iteration or moved to the project backlog if they can only be remedied during the next iteration or release cycle.

Our understanding of the agile philosophy, values, principles, and the mechanics of implementation of agile methods and processes such as the generic process described above have guided the identification of the objectives, principles, strategies, and properties which are the core components of the OPS Framework.

## *2.2 Agile Assessment*

The agile manifesto states that agile practitioners value "working software over comprehensive documentation" [5]. This value, in conjunction with the agile principle that states "our highest priority is to satisfy the customer through early and continuous delivery of valuable software" [5], have been the mantra for guiding extant agile assessment procedures and supporting metrics [20]. Consequently, *most assessment approaches have focused primarily on the product*. For example, the number of bugs reported [21], the number of tests written for maximum code coverage [22, 23], the number of broken builds [23], the team velocity that indicates the number of story points delivered during each iteration [20], earned business value [11], etc. are typical product metrics used by agile teams.

While essential to assessing agile methods, product metrics alone are insufficient to support a *comprehensive* approach to assessment. Software Engineering involves *people, process, project,* and *product* (the 4 P's) [24]. Hence, metrics used in the assessment of agile methods should incorporate characteristics of the 4P's. The OPS Framework presented in this dissertation is designed to incorporate the 4P's. In this section, we review some of the current agile assessment approaches that have motivated our research.

### 2.2.1 Agile Assessment Checklists

Agile practitioners are aware that in order to continuously improve an adopted agile method, it is necessary to assess both their process and the final product. As a rule, agile teams are constantly being asked: *"How agile are you?"* In effect, teams need to determine the extent to which their process is agile. To assess the agility of their process, many teams rely on checklists to determine the presence or absence of practices that are considered "agile". Some checklists commonly used by agile practitioners are (1) the *Nokia Test* [25] *for Scrum*, (2) *How Agile Are You (42-Point Test)* [26], (3) the *Scrum Master Checklist*

[27], and (4) the *Do It Yourself (DIY) Project Process Evaluation Kit* [28]. The DIY Project process Evaluation Kit is a generic checklist that can be adapted for use with any agile process. These checklists, however, focus primarily on practices, and largely ignore the underlying agile principles and objectives, as well as the effectiveness of an agile method. Moreover, most of these checklists are tailored to one or more specific agile methods.

### 2.2.2 Agile Adoption Frameworks

With the increasing popularity of agile methods, more and more organizations are moving towards agility. However, researchers have realized that many organizations and teams adopt and use agile practices without a proper understanding of the agile philosophy. Moreover, many organizations proclaim themselves to be 'agile' when in fact they are following what is considered "Agilefall" (using agile terms but essentially following the waterfall model) [25]. In order to mitigate this issue and guide organizations in their agile adoption efforts, several agile adoption and process improvement frameworks have been developed.

In his book titled "Balancing Agility and Discipline – A guide for the perplexed", Barry Boehm provides a five step risk-based software development approach that is reflective of the characteristics of both agile and plan-driven development methodologies [29]. This framework is helpful for organizations that require a hybrid approach combining the best aspects of agile and plan-driven software development methods. However, the primary disadvantage of this approach is that the framework provides an overview of the method to be followed, but lacks an enunciation of the actual practices that should be used [30].

Organizations require a tangible approach to adopting agile methods. More specifically, before transitioning to agility, they need specific guidance on which practices are best suited for their requirements. Agile process improvement frameworks such as the Sidky Agile Measurement Index (SAMI) [30, 31] and Agile Adoption and Improvement Model (AAIM) [32] guide an organization's agile adoption and improvement efforts. Both frameworks describe levels of agility modeled on similar concepts found in the Software Capability Maturity Model (SW CMM) [33] and Capability Maturity Model Integration (CMMI) [34]. That is, a set of practices is to be adopted at successively higher levels in order to be 'agile' at that level. The primary disadvantage of these frameworks is that a set of practices is 'forced' on an organization at defined levels, which compromises the flexibility offered by agile methods. We do recognize, however, that these approaches are not intended to assess the 'goodness' of an agile method, but instead, to guide an organization's transition toward an agile software development paradigm. In the next paragraphs, we discuss the SAMI and the AAIM approaches to agile adoption and improvement. Both frameworks specify practices that an organization should implement in order to embrace agility.

***Sidky Agile Measurement Index (SAMI)***

SAMI was developed at Virginia Tech by Dr. Ahmed Sidky as an attempt to guide the agile adoption efforts of an organization [30]. The SAMI is based on the idea that the more the number of agile practices adopted by an organization, the greater its agility [30]. The number of agile practices adopted by an organization determines its *agile potential.* Recognizing that counting the number of practices adopted is a simplistic measure, Dr. Sidky has created five agile levels. The agile levels defined in SAMI are very similar to the levels of maturity in CMM and CMMI.

Each agile level has a set of objectives. These objectives reflect the focal values stated in the agile manifesto. Each level is associated with a set of principles stated in the manifesto and practices followed by the agile community [30]. At each level, a set of practices that are reflective of the objectives defined at that level have been identified.

*An organization can choose to achieve any of the specified levels of agility by adopting all the practices at that level and the levels below***.** That is, for an organization to maintain their agility at level 3, all the practices identified for levels 1, 2 and 3 have to be adopted.

SAMI outlines a systematic and structured approach to guide organizations in their agile adoption efforts. The primary disadvantage of the SAMI is that it compromises the flexibility afforded by agile methods. A set of practices for each level is predefined and is "forced" on the organization and thus reducing the flexibility offered by agile methods. Moreover, the established levels may not be reflective of the culture and values of the organizations.

***Agile Adoption and Improvement Model (AAIM)***

Qumer's AAIM [32] is very similar to SAMI in that it provides an agile adoption and improvement framework that specifies varying levels of agility. The intent of AAIM is to measure the degree of agility of an agile method. Agility is defined by five parameters, namely "flexibility, speed, leanness, responsiveness and learning" [35].

Six levels of agility are defined and are further grouped into three agile blocks. Each block and level has an associated set of objectives. At each block, the degree of agility can be measured quantitatively using the agility measurement modeling approach – the 4-DAT (4 Dimensional Analytical Tool) [35, 36].

The 4-DAT has been designed to examine agile methods from four different dimensions: (1) determining the scope of application of the method, (2) agility characterization based on the five parameters mentioned previously - flexibility, speed, leanness, responsiveness and learning, (3) value-based characterization – identifying practices based on the focal values stated in the agile manifesto, and (4) software process characterization – identifying practices covering the SDLC phases, project management, configuration management, and process management [35].

Dimensions 1, 3 and 4 involve a qualitative assessment; dimension 2 involves a quantitative assessment. In dimension 2, the presence or absence of the practices is recorded and the overall score serves as a measure to check the existence of agility in agile methods.

AAIM is modeled along similar lines as the SAMI with respect to the CMMI-like agile levels. As mentioned earlier, the disadvantage is reduced flexibility. The degrees of agility are measured by analyzing the adoption of a set of practices. Also, the predefined levels may not be 'in-sync' with the organizational objectives. Additionally, the AAIM and SAMI do not provide an indication of the effectiveness of the agile approach under consideration. However, we recognize that these Frameworks are intended to guide agile adoption efforts and are not designed for post-adoption agile assessment.

## 2.2.3 Agility Measurement Approaches

After transitioning to agility, most organizations are concerned about how 'good' has their agile adoption been - that is, *are they achieving what they set out to by adopting the agile philosophy?* Also, the organizations are interested in identifying problem areas and issues, and take adequate measures to solve them. Retrospective meetings at the end of each iteration or release cycle help an organization or team assess their progress, and 'fine-tune' their agile approach. In addition to retrospection, teams can employ external consultants or tools to help assess their agile process. In this sub-section, we discuss three post-adoption agile assessment tools – Comparative Agility [37, 38], ThoughtWorks Agile Assessment [39], and the Agile Usage Model [40].

***Comparative Agility***

Some organizations focus on being more agile than their competition rather than striving to be "perfectly agile" [37]. The Comparative Agility (CA) assessment tool (developed by Kenny Rubin, Mike Cohn, and Dr. Laurie Williams) is used to help organizations assess their agility relative to other organizations or teams that responded to the tool [37]. CA is a survey-based assessment tool. "Any agile practitioner can visit the CA website <http://comparativeagility.com/>, answer the survey questions, and receive a free report that compares their results to the complete industry dataset" [37]. The answers are recorded on a five point Likert scale. Additionally, practitioners could request a customized survey.

At the highest level, CA identifies seven dimensions that form the basis for the agility assessment. These dimensions can be likened to the Key Process Areas (KPAs) identified by the CMM and CMMI. The seven dimensions are: teamwork, requirements, planning, technical practices, quality, culture and knowledge creating [37, 38]. Each dimension is composed of three to six characteristics. Each characteristic has four statements that are assessed by the survey respondents [37]. "Each statement is a practice for which the respondent indicates the truth of the statement relative to their team or organization" [37]. By utilizing a combination of dimensions, characteristics, and statements, a team or an organization, can gauge their agility relative to that of their competitors or themselves at an earlier time.

Through close analysis of the formulated survey questions (available from [38]), it is evident that the statements require an indication of presence or absence of a practice rather than how well is it being used. Respondents base their answers on their observations that are not validated by empirical data. Also, the answers to the survey questions are subjective. Moreover, when comparing the agility of two or more organizations, it is unclear if the tool factors in the differences in their organizational objectives.

### *ThoughtWorks Agile Assessment*

ThoughtWorks is a leading agile development and consulting company. They have developed an agile assessment survey which is available on their website <http://www.agileassessments.com/>. Agile practitioners can complete the survey to get a report on the level of agility within their organization or team, and also identify opportunities for improvement [41]. The survey is composed of twenty questions covering development and management practices [39, 41].

The ThoughtWorks Agile Assessment survey questions are intended to gather information about the existence and usage of the practices. This approach, like CA, does not address assessing the effectiveness of agile methods in the survey. However, the creators of the tool recognize and state that the survey is intended to offer *preliminary* insights into the agile method being followed by a team or organization and is not a replacement for a customized assessment approach.

### *Agile Usage Model*

The Agile Usage Model [40] has been designed to measure the post-implementation effectiveness of agile methods. This model views "effectiveness as an outcome of increased agile usage"[40]. Agile effectiveness is measured based on productivity in and quality of the development process, and customer satisfaction. Although this approach recognizes that existing agile assessment processes do not assess the effectiveness of agile methods, and attempts to mitigate this problem, it does not assess the extent to which an adopted agile method helps achieve organizational objectives or the degree to which the organization is capable of supporting the implementation of that method.

### 2.2.4 Summary

The checklists, the agile process improvement frameworks, and the agile assessment approaches discussed in this section are all distinct tools. Individually, each assesses in part, different aspects of an agile method. They are, however, somewhat limited in scope and application. Hence, there is a need for a more comprehensive approach to assessing agile methods. We present three criteria or perspectives for assessing the 'goodness' of agile methods, namely adequacy, capability, and effectiveness. The OPS Framework guides assessment based on three perspectives, which combine to provide a more comprehensive approach. We discuss the OPS Framework, its components, and our approach to assessing capability in Chapter 3.

## *2.3 Guiding Frameworks*

The OPS Framework and the assessment approach developed in this research have been guided by the Objectives, Principles and Attributes (OPA) Framework and the Evaluation Environment (EE) Methodology. We present these two approaches in this sub-section. We also preface this discussion with an overview of some of the earlier software measurement approaches.

### 2.3.1 Early Work in Software Measurement

Software Metrics have been in use since the mid 1960's when the *Lines of Code* metric was adopted as the basis for measuring programming productivity and effort [42]. The rationale for the definition and use of any individual software metric has been either:

> "(a) the desire to assess or predict effort/cost of development processes, or
> (b) the desire to assess or predict quality of software products" [42].

Hence, most Software Measurement approaches adopted by organizations utilize metrics to assess or predict the quality of software products. Metrics to assess quality involve the assessment of software quality objectives such as maintainability, reliability, testability, correctness, adaptability, etc.

The primary goal of Software Engineering is to produce a quality product [43]. The abstract nature of software has contributed to the difficulties in assessing its quality. Consider the following examples discussed in [44] and [45] respectively:

1. If the goal is to assess maintainability, we may conjecture that as the number of unconditional branches in a program increases, the more difficult it would be to maintain the program.

2. Schedule slippage adversely impacts the quality of the software product. However, one might incorrectly surmise that if a project is on schedule, the system being built is a quality product.

In both cases mentioned above, the measures lack the definitive linkage that directly relates the measure to software quality. That is, the measures discussed above are marginally related to the goal. Hence, predicting the software quality using such measures that are "somewhat" related to the goal is difficult. Recognizing the need for defined measures to be directly related to software quality, researchers developed metrics and measurement programs that utilized attributes of the process to assess the product. McCall's Factor/ Criteria/ Metric [44], Goal Question Metric (GQM) [46], the Objectives Principles and Attributes (OPA) Framework [45, 47, 48], and the Evaluation Environment (EE) Methodology[49-52] are a few assessment approaches that have been developed to address the above- mentioned issue.

Consider the "Goal Question Metric" (GQM) method. "Goal-Question-Metric (GQM) is a paradigm for the systematic definition, establishment, and exploitation of measurement programs supporting the quantitative evaluation of software processes and products." [53].

The GQM method defines a measurement model consisting of three levels [46]:

- Conceptual level **(GOAL) -** A goal defined for an ***object*** *(that part of the product or process being observed)* based on entities like

  a) ***purpose*** *( the motivations for studying the object)*,
  b) ***issue*** *(the characteristic of the object under study)* and
  c) ***viewpoint*** *(people interested in studying the object)*

- Operational level **(QUESTION) -** A set of questions formulated that must be answered to determine if the goals have been reached.

- Quantitative level **(METRIC) -** A set of metrics derived from the questions to collect data to answer the same quantitatively. The collected data could be objective or subjective.

The GQM approach to goal-oriented measurement consists of a *top-down decomposition or refinement* of the goals into questions and then into metrics and a *bottom-up analysis and interpretation* of the collected data [54]. The data collected by the use of metrics are used to answer the formulated questions, which in turn determine if the stated goals have been met. Here, there is an established relationship between the goal, the questions and the metrics. Hence, we can definitively predict and/or assess quality characteristics using metrics that have been formulated to address specific questions regarding those characteristics.

### 2.3.2 Objectives Principles Attributes (OPA) Framework

The OPS Framework that we propose for the assessment of the 'goodness' of agile methods is based on the OPA Framework. The OPA Framework (developed at Virginia Tech) is a structured and systematic approach that serves as a basis for the evaluation of Software Engineering Methodologies [47]. The OPA Framework identifies:

- Project-level software engineering ***objectives*** like maintainability, reliability, correctness, etc. that are commonly recognized by various methodologies
- ***Principles*** that are reflective of the software development process such as information hiding, structured programming, etc, and
- ***Attributes*** that result from the utilization of the said principles such as readability, traceability, etc. [45].

"Achievement of the objectives comes through the application of principles supported by a methodology. Employment of the said principles result in software products possessing attributes considered desirable and beneficial" [47]. Also, definitive linkages between these components are defined. In addition to the above-mentioned components, the framework identifies properties that are observable characteristics of the product. These properties are indicative of the extent to which the product possesses the desirable attributes. Each attribute is linked to one or more properties. Each attribute, property pair forms an indicator. The indicators form the basis for defining metrics. The values obtained by the usage of these metrics attest to the existence of desirable software engineering attributes in the product [47]. These values are propagated and aggregated at the principles level to present an assessment of the development process, and then further aggregated at the objectives level to suggest the extent to which the objectives are being achieved.

A Software Engineering methodology is assessed from two different perspectives – adequacy and effectiveness. Adequacy denotes the extent to which a methodology can support the achievement of stated project-level goals and objectives [47]. Assessing the adequacy involves a top-down examination of the linkages between the objectives, principles, and attributes to attest to the existence of those Framework components. The evaluation of the adequacy of a methodology is intended to reveal its deficiencies [47]. While a top-down traversal of the linkages is adopted for assessing the adequacy of a methodology, the effectiveness of that methodology is determined by a top-down and bottom-up examination through the linkages [47]. "Effectiveness of a methodology can be defined as the degree to which that methodology produces the desired results" [47]. Analyzing the metric values imply the degree to which particular properties are observed. In turn, this information can be used to identify a set of attributes present. If these attributes are different from the set of attributes identified during the adequacy assessment, it can be implied that the users have failed to adhere to the principles supported by the methodology [47].

## 2.3.3 Evaluation Environment Methodology

Evaluation of system of systems is becoming increasingly difficult because the systems themselves are large and complex. Moreover, the process involves the measurement and evaluation of hundreds of qualitative and quantitative elements, mandates evaluations by Subject Matter Experts (SMEs), and requires the integration of disparate measurements and evaluations [49]. Hence, "planning and managing such evaluation efforts require a unifying methodology and should not be performed in an ad-hoc manner" [49]. In order to address the issues in evaluating large and complex systems, the Evaluation Environment (EE) methodology has been developed [49, 51, 52] at Virginia Tech by Dr. Osman Balci. The EE methodology has been designed to conduct evaluations on different types of projects classified under various domains. In order to accommodate geographically distributed teams, a Web-based client/server software system has also been developed and used for complex DoD evaluation projects. The EE Methodology mandates evaluations by SMEs. Some of the main characteristics of the EE Methodology that are adopted by the OPS Framework are presented below [50]:

1. ***Indicators*** - "Qualitative concepts" [49] such as maintainability, reliability, correctness, etc. cannot be measured directly. The EE methodology proposes to use a hierarchy of indicators to measure quality attributes. Each "qualitative concept" that cannot be measured directly is a *root indicator* [49]. Hence, to measure each "qualitative concept", we decompose the root indicator into multiple levels of child indicators such that the indicators at the lowest level can be assessed directly. The indicators at the lowest level are called the *leaf indicators*. *Branch indicators* are those that have a parent indicator and at least one child indicator. We assess only the leaf indicators directly.

2. ***Decomposition*** – As mentioned in the previous paragraph, we decompose a root indicator into the branch and leaf indicators. The multiple levels of indicators form a hierarchy that is an acyclic graph. Also, "each leaf indicator is manageable in complexity and is directly assessable by way of testing, direct measurement, analysis, or examination" [50]. Only the leaf indicators are measured and evaluated.

3. ***Integration*** - When there are multiple leaf indicators for a root indicator, the SMEs critically weigh the leaf indicators by performing a pair-wise comparison using the Analytical Hierarchy Process

(AHP). More often than not, the leaf indicators are assigned equal weights. Leaf indicators are evaluated by a variety of techniques such as testing, direct measurement, analysis, and examination. All the evaluations are then integrated under a structured framework. Leaf indicator evaluation scores are integrated in a bottom-up fashion based on the criticality weightings of the branch indicators. The integration results in an overall evaluation score for the root indicator [49, 50]. This final score would help us evaluate the root indicator.

### 2.3.4 Relationship of the OPA Framework and EE Methodology to the OPS Framework

The Objectives, Principles, and Attributes (OPA) Framework [48] and the Evaluation Environment (EE) Methodology [51] guide the structure of the OPS Framework and the assessment approach developed in this research. Both approaches present hierarchical assessment frameworks where qualitative concepts such maintainability, reliability, correctness, etc. are evaluated by using defined measures that are directly related to those concepts. We adopt the hierarchical structure touted by both frameworks to assess the extent to which an agile method achieves its stated objectives. The OPA Framework provides the concept of objectives, principles, strategies, and linkages and the assessment perspectives of adequacy and effectiveness. The EE Methodology serves to guide the development of our indicator hierarchy, indicator scoring, and score aggregation. Similar to both Methodologies, our assessment of capability and effectiveness involve the evaluation of people, process, project, and product properties.

## References

[1] *IEEE Standards Collection: Software Engineering* (1993)*,* IEEE Standard 610.12-1990.

[2] W. W. Royce (1987), "Managing the Development of Large Software Systems: Concepts and Techniques," presented at the Proceedings of the 9th international conference on Software Engineering, Monterey, California, United States.

[3] B. Boehm (1988), "A Spiral Model of Software Development and Enhancement," *Computer,* vol. 21, pp. 61-72.

[4] R. S. Pressman (2005), *Software Engineering: A Practitioner's Approach*, 6 ed.: McGraw - Hill International Edition.

[5] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn*, et al.* (2001), *Manifesto for Agile Software Development*, http://www.agilemanifesto.org

[6] J. Patton (2008), "Ambiguous Business Value Harms Software Products," *IEEE Software,* vol. 25, pp. 50-51.

[7] K. Beck and C. Andres (2004), *Extreme Programming Explained: Embrace Change*, 2 ed.: Addison-Wesley Professional.

[8] R. Jeffries (2002), *What Is Extreme Programming*, http://xprogramming.com/what-is-extreme-programming

[9] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta (2002), *Agile Software Development Methods: Review and Analysis*. Finland: VTT Publications.

[10] K. Schwaber and J. Sutherland (2011), *The Scrum Guide*, http://www.scrum.org/Portals/0/Documents/Scrum Guides/Scrum_Guide.pdf - zoom=100

[11] A. Cockburn (2004), *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Professional.

[12] A. Cockburn (2002), *Agile Software Development*. Addison-Wesley.

[13] S. R. Palmer and J. M. Felsing (2002), *A Practical Guide to Feature Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR.

[14] M. Poppendieck and T. Poppendieck (2003), *Lean Software Development: An Agile Toolkit*. Addison Wesley.

[15] M. Poppendieck (2003), Lean Software Development. *C++ Magazine Methodology Issue*.

[16] G. Smith and A. Sidky (2009), *Becoming Agile in an Imperfect World*. Greenwich, CT: Manning Publications Co.

[17] D. Leffingwell (2007), *Scaling Software Agility: Best Practices for Large Enterprises*, 1 ed.: Addison-Wesley Professional.

[18] J. Highsmith (2002), *Agile Software Development Ecosystems*. Addison-Wesley.

[19] R. C. Martin (2003), *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR.

[20] D. Nicolette (2009), *Agile Metrics*, http://www.slideshare.net/rsrivastava91/agile-metrics-v6

[21] R. Petit (2006), Agile Processes: Making Metrics Simple, Agile Journal.

[22] A. Ruiz (2010), *Effective Code Coverage (and Metrics in General)*, http://alexruiz.developerblogs.com/?p=1421

[23] A. Janus, A. Schmietendorf, R. Dumke, and J. Jager (2012), "The 3c Approach for Agile Quality Assurance," in *3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)*, pp. 9-13.

[24] E. J. Braude (2001), *Software Engineering: An Object-Oriented Perspective*. John Wiley & Sons, Inc.

[25] J. Little (2007), *The Nokia Test,* http://agileconsortium.blogspot.com/2007/12/nokia-test.html

[26] K. Waters (2008), *How Agile Are You? (Take This 42 Point Test)*, http://www.allaboutagile.com/how-agile-are-you-take-this-42-point-test/

[27] M. James (2007), *A Scrummaster's Checklist*, http://blogs.danube.com/a-scrummasters-checklist?q=blog/michaeljames/a_scrummasters_checklist

[28] G. Dinwiddie (2009), *Diy Project/Process Evaluation Kit*, http://blog.gdinwiddie.com/2009/08/18/diy-projectprocess-evaluation-kit/

[29] B. Boehm and R. Turner (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.

[30] A. Sidky (2007), "A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework," Dissertation, Computer Science, Virginia Tech, Blacksburg, VA.

[31] A. Sidky, J. D. Arthur, and S. Bohner (2007), "A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework.," *Innovations in Systems and Software Engineering,* vol. 3, pp. 203-216.

[32] A. Qumer, B. Henderson-Sellers, and T. McBride (2007), "Agile Adoption and Improvement Model," in *European, Mediterranean & Middle Eastern Conference on Information Systems (EMCIS)*, Polytechnic University of Valencia, Spain.

[33] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber (1993), "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute.

[34] W. Royce (2002), CMM vs. CMMI: From Conventional to Modern Software Management. *Rational Edge*.

[35] A. Qumer and B. Henderson-Sellers (2008), "An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering," *Information and Software Technology,* vol. 50, pp. 280-295.

[36] A. Qumer and B. Henderson-Sellers (2008), "A Framework to Support the Evaluation, Adoption and Improvement of Agile Methods in Practice," *Journal of Systems and Software,* vol. 81, pp. 1899-1919.

[37] L. Williams, K. Rubin, and M. Cohn (2010), "Driving Process Improvement Via Comparative Agility Assessment," in *AGILE Conference, 2010*, pp. 3-10.

[38] L. Williams, K. Rubin, and M. Cohn (2013), *Comparative Agility*, http://comparativeagility.com/

[39] Thoughtworks (2013), *Agile Assessments*, http://www.agileassessments.com/

[40] M. Senapathi and A. Srinivasan (2012), "Understanding Post-Adoptive Agile Usage: An Exploratory Cross-Case Analysis," *Journal of Systems and Software,* vol. 85, pp. 1255-1268.

[41] Thoughtworks (2009), "Agile Self Evaluation Version 1.0 Prepared for Ciprian Mester, Alcatel-Lucent."

[42] N. E. Fenton and M. Neil (2000), "Software Metrics: Roadmap," presented at the Proceedings of the Conference on The Future of Software Engineering.

[43] S. D. Conte, H. E. Dunsmore, and V. Y. Shen (1986), *Software Engineering Metrics and Models*. Benjamin/Cmmings Publishing Company, Inc.

[44] J. P. Cavano and J. A. McCall (1978), "A Framework for the Measurement of Software Quality," presented at the Proceedings of the software quality assurance workshop on Functional and performance issues.

[45] R. E. Nance and J. D. Arthur (2002), *Managing Software Quality: A Measurement Framework for Assessment and Prediction*. Springer.

[46] V. R. Basili, G. Caldiera, and H. D. Rombach (1994), "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc.

[47] J. D. Arthur, R. E. Nance, and S. M. Henry (1986), "A Procedural Approach to Evaluating Software Development Methodologies: The Foundation," Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

[48] J. D. Arthur and R. E. Nance (1991), "A Framework for Assessing the Adequacy and Effectiveness of Software Development Methodologies," Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

[49] O. Balci, R. J. Adams, D. S. Myers, and R. E. Nance (2002), "Credibility Assessment: A Collaborative Evaluation Environment for Credibility Assessment of Modeling and Simulation Applications," presented at the Proceedings of the 34th conference on Winter simulation: exploring new frontiers, San Diego, California.

[50] O. Balci (2008), "Collaborative Evaluation Environment: Methodology and Web-Based Software," CS 6704 Class Notes, Fall 2008, Department of Computer Science, Virginia Tech, Blacksburg, VA.

[51] O. Balci and W. F. Ormsby (2008), "Network-Centric Military System Architecture Assessment Methodology," *International Journal of System of Systems Engineering,* vol. 1, pp. 271-292.

[52] M. L. Talbert (1995), "A Methodology for the Measurement and Evaluation of Complex System Designs," Dissertation, Computer Science, Virginia Tech, Blacksburg, VA.

[53] A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti*, et al.* (1998), "Applying Gqm in an Industrial Software Factory," *ACM Transactions on Software Engineering and Methodology,* vol. Vol. 7,

[54] F. van Latum, R. van Solingen, M. Oivo, B. Hoisl*, et al.* (1998), Adopting GQM-Based Measurement in an Industrial Environment. *IEEE Software*.

# 3 The OPS Framework and Assessment Methodology

Our research is motivated by the lack of a comprehensive approach to assessing agile methods. We assess the collective ‘goodness’ of an agile method based on (1) its *adequacy*, (2) the *capability* of the organization to provide the supporting environment to implement the method, and (3) the method’s *effectiveness*. We define adequacy, capability and effectiveness as below (definitions adapted for current context from [1, 2]):

- *Adequacy* - Sufficiency of the method with respect to meeting its stated objectives.
- *Capability* – Ability of an organization to provide an environment supporting the implementation of its adopted method. Such ability is reflected in the characteristics of an organization's people, process and project.
- *Effectiveness* – Producing the intended or expected results. The existence of necessary process artifacts and product characteristics indicate levels of effectiveness.

We have designed the **O**bjectives **P**rinciples and **S**trategies (**OPS**) Framework to assess that ‘goodness’. In this chapter, we present the core components of the OPS Framework and the approach to assessing adequacy, capability, and effectiveness.

The agile manifesto provides four focal values and twelve principles that define the agile philosophy. Using the manifesto as a guide, we have evolved the OPS Framework (see Figure 3.1) to reflect the viewpoint that each agile method should

- strive to achieve an enunciated set of desirable *objectives*,
- embrace process *principles* that support the achievement of those objectives, and
- employ accepted agile *strategies* to implement those principles.

In addition to the objectives, principles, and strategies, the Framework also identifies definitive linkages that establish the relationships between these core components. The linkages are fundamental to our assessment approach. They bind specific principles to the achievement of an individual objective and strategies to each principle they implement. In Figure 3.1, the arrows between the objectives, principles, and the strategies depict the linkages. As mentioned previously, to assess adequacy, we follow the linkages in a top-down fashion. To assess capability and effectiveness, we perform both a top-down and bottom-up traversal of the linkages. Additionally, the bottom-up assessment process entails the identification of observable properties of the people, process, project, product, and the environment. As shown in Figure 3.1, these properties are associated with the strategies. Each (strategy, property) pair forms an indicator. Determining the extent to which these properties are observed is the first step in the approach to assessing capability and effectiveness (see Section 3.2).

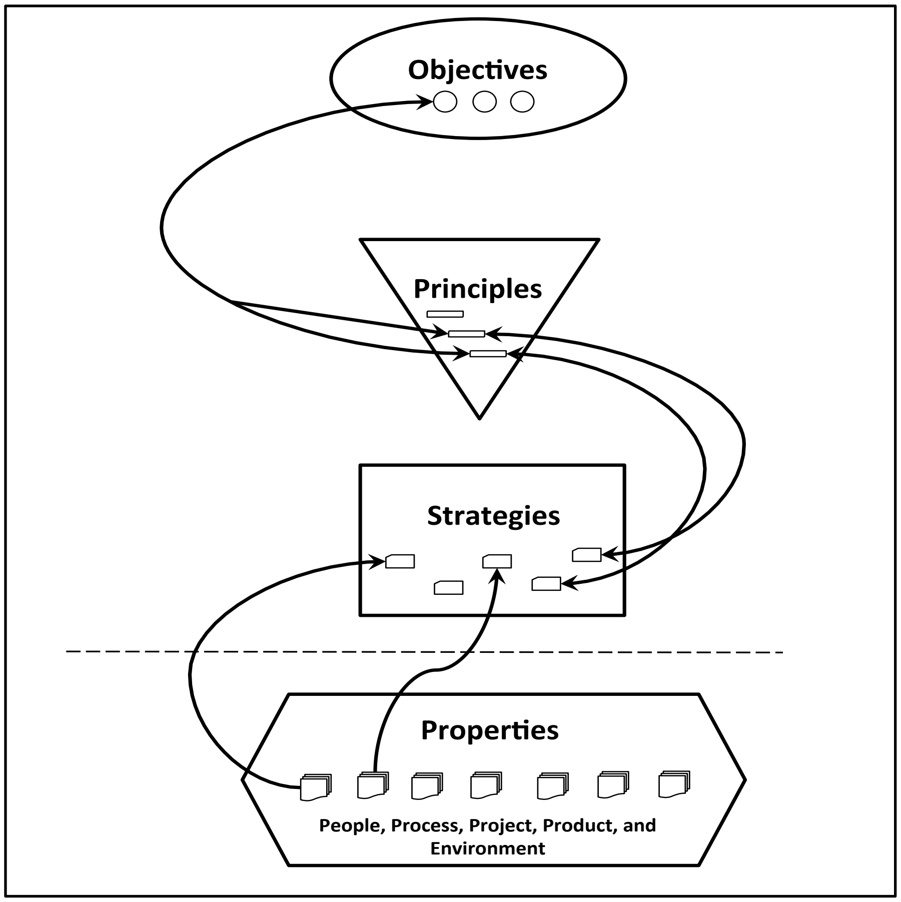


*Figure 3.1. OPS Framework*

## 3.1 The OPS Framework

The design and construction of the OPS Framework is based on the recognition that any viable agile method strives to achieve a set of desired objectives, asserts principles that govern the achievement of those objectives, and includes strategies to implement those principles. In concert with the above, the OPS Framework identifies ***five objectives*** that are reflective of the agile philosophy, ***nine principles*** that govern the development process supporting the achievement of those objectives, and ***17 strategies*** that help implement those principles (see Figure 3.2).

### 3.1.1 Objectives

An objective is something aimed at or striven for [2]. We contend that each agile method embodies a set of objectives, which are supported by a set of underlying principles. The OPS Framework identifies five objectives that are reflective of the agile philosophy (Figure 3.2). These objectives provide a more definitive description of the values articulated in the manifesto. They are common themes that underlie agile software development methods and have been identified from sources including, but not limited to, the agile manifesto [3], books [4-9], research papers [10, 11], experience reports, white papers [12] and discussions with industry experts.

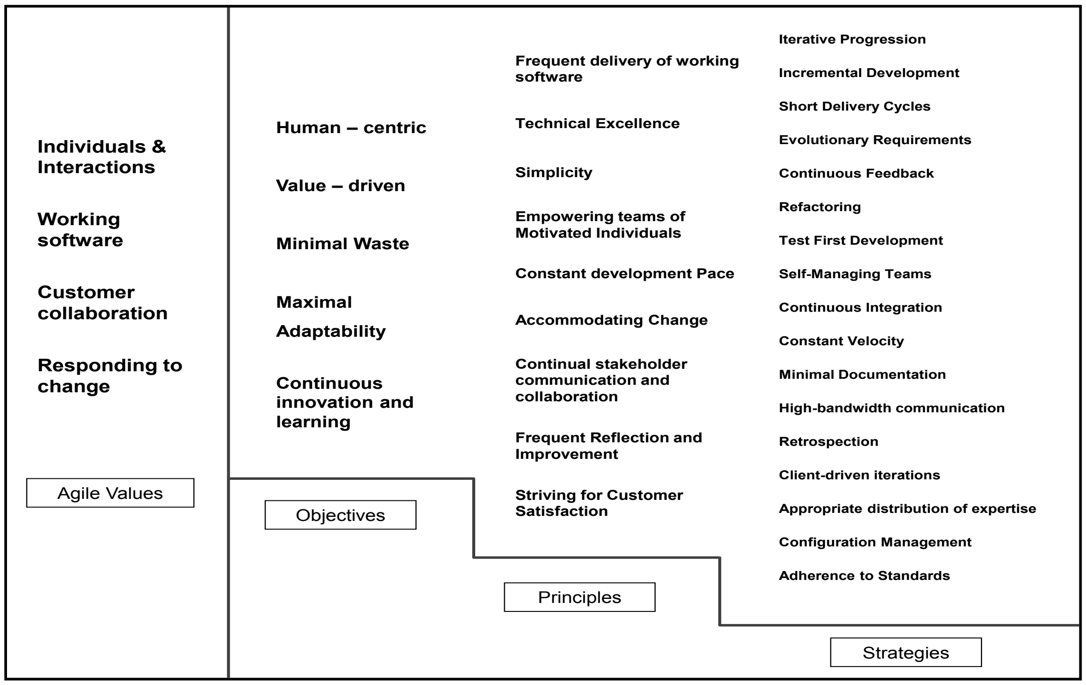


***Figure 3.2. Objectives, Principles, and Strategies identified by the OPS Framework***

As shown in Figure 3.2, the identified objectives are human centric, value driven, minimal waste, maximal adaptability and continuous innovation and learning. These objectives are consistent with the common themes of agile software development identified by Jim Shore in [5]. The working definitions of the objectives can be found in Table 3.1 and Appendix A. We briefly discuss the five objectives identified by the Framework in the following paragraphs.

*Human centric*

The agile philosophy places utmost importance on people. The first focal value in the manifesto states, "people are more important than processes and tools." Hence, teams adopting agile methods propose open environments for working, encourage face-to-face communication among the stakeholders (stakeholders include customers, users, developers, project managers, business analysts, testers, etc.) and provide support to the team members. The stakeholders work together throughout the project. In short, the agile movement focuses on fostering team spirit among the members of the software development team by emphasizing on close team relationships and close working environments. Human aspects of the software development process are considered more important than the process itself. The working definitions of the identified objectives are provided in Table 3.1 and in Appendix A.

*Value driven*

The agile manifesto states that "working software" is more valuable to the customers than "comprehensive documentation". From this statement we can infer that producing quality software, on time and within budget is of utmost importance. We also recognize that value is something that can be derived from any software or service that "delivers profit to the organization, paying for the software in the form of an Increase in Revenue, an Avoidance of Costs, or an Improvement in Service (IRACIS)" [13]. Hence, 'value driven' (see Table 3.1, Appendix A) is one of the more important objectives of the agile software development paradigm.

*Minimal Waste*

Agile methods focus on developing quality software on time, within budget, and of value to the customers. These methods advocate iterative and incremental development processes. Additionally, software or software increments are delivered frequently at regular intervals. Delivering valuable software frequently entails building only what is necessary. Studies show that more often than not, only 20% of the features included in a software system are used 80% of the times [14]. Recognizing that customers and users primarily utilize only a subset of the developed features, it is prudent to minimize waste by understanding customer and user needs and building only the feature sets that are necessary. Additionally, we realize that frequent delivery of valuable software requires the process overhead to be minimal. That is, the development process should be kept simple. The concept of minimizing waste is not explicitly stated as a focal value in the manifesto. It is however included in the set of identified principles. A common theme underlying agile methods is the notion of being “lean.” Hence, the OPS Framework identifies the concept of ‘Minimal Waste’ as an objective of the agile philosophy.

*Maximal adaptability*

Consider the agile value “responding to change over following a plan”. We assert that responding to change is not limited to accommodating changes to the elicited features, but also to change in the process so as to better meet the needs of the organizations and the teams. For example, let us assume that a team involved in agile software development decides to implement Test Driven Development (TDD) [15]. If the members later realize that they do not possess the required skills to effectively implement TDD, they should have the opportunity to either acquire that expertise, or the freedom to change the process by adopting a different strategy without compromising the effectiveness and efficiency of the process. The objective ‘Maximal Adaptability’ is reflective of the above-mentioned focal value. The objective and its working definition (see Table 3.1 and Appendix A), present the manifesto value in a more definitive way.

*Continuous Innovation and Learning*

We recognize that agile values and principles have evolved since the creation of the manifesto in 2001. Hence, the objectives are intended to reflect the current state of agile software development. For example, most agile methods express strategies that embrace the notion of continuous process improvement. That is, practitioners are encouraged to re-examine their processes at regular intervals to ensure that the process remains “efficient and effective” [9]. Also, based on their experiences, practitioners “fine-tune” their adopted agile methods to best suit their needs. This is an important concept that is supported by three principles enunciated in the manifesto – technical excellence, simplicity and retrospection [9]. However, the manifesto does not explicitly state “continuous process improvement” as a focal value. The objectives identified by the OPS Framework reflect both stated and unstated agile values. Hence, ‘Continuous Innovation and Learning’ is stated as one of the five objectives of the agile philosophy.

***Table 3.1. Working definitions of the identified Objectives***

| **Objectives** | |
|---|---|
| ***Human – centric*** | People are more important than processes, practices and tools |
| ***Value-driven*** | Maximize stakeholder value(s): increased revenue, improved customer satisfaction, reduced cost, etcetera. |
| ***Minimal Waste*** | Keep things simple - build only what is necessary. |
| ***Maximal Adaptability*** | Maintain flexibility: (a) accommodate change and (b) freedom to choose appropriate practices |
| ***Continuous innovation and learning*** | Innovate and improve the development process through the frequent examination and evaluation of past development activities. |

Table 3.1 provides the working definitions of the five objectives identified by the OPS Framework (also found in Appendix A). These objectives are consistent with practitioners' views of the common themes underlying agile software development methods (see Chapter 4).

## 3.1.2 Principles

Principles govern the process by which one achieves the desired objectives [2]. The agile manifesto presents twelve principles that underlie the agile software development process. To a large extent, the principles outlined in the manifesto overlap to preserve the different styles offered by the various agile methods. However, in order to provide a more concise view of the agile software development paradigm, the OPS Framework embodies a complementary set of nine principles (Figure 3.2).

Consider the objective 'Maximal Adaptability' discussed in the previous section. Based on our working definition for Maximal Adaptability ('*maintaining (a) the flexibility to support change and (b) the freedom to choose among appropriate strategies*' as outlined in Appendix A), we conjecture that the following four principles govern the achievement of the said objective (see Figure 3.3):

- *Frequent delivery of working software*

  Delivering software frequently and at regular intervals facilitates gathering early feedback from the customers. Hence, changes can be identified and addressed before major portions of the system are built.

- *Simplicity*

  Building only what is necessary enables the team to keep the architecture and design of the system simple. This reduced complexity also helps facilitate change in an easier and faster manner.

- *Accommodating change*

  It is evident that this principle is directly related to the objective of 'maximal adaptability'. It indicates that the process of accommodating change should be achieved with minimal impact on other parts of the system.

- *Frequent reflection and learning*

  Continuous process improvement is also one of the fundamental concepts of the agile philosophy. Frequently reflecting on the process and learning from past development efforts helps a team in choosing the right strategies and principles that are the best suited for project needs.

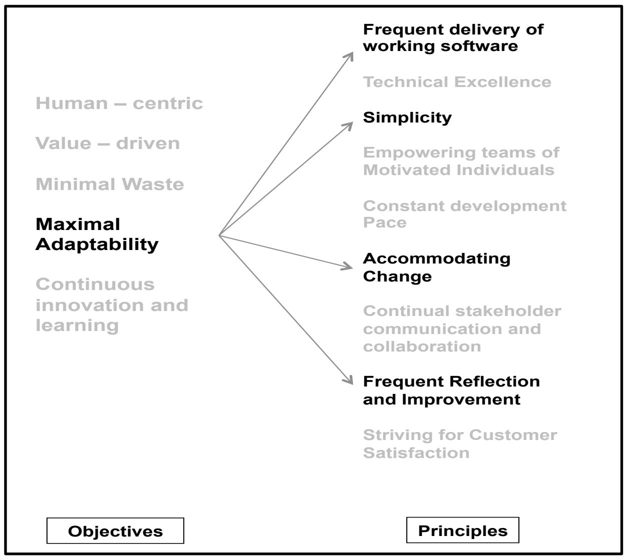

Human – centric
Value – driven
Minimal Waste
Maximal Adaptability
Continuous innovation and learning
Frequent delivery of working software
Technical Excellence
Simplicity
Empowering teams of Motivated Individuals
Constant development Pace
Accommodating Change
Continual stakeholder communication and collaboration
Frequent Reflection and Improvement
Striving for Customer Satisfaction
Objectives
Principles

***Figure 3.3. Example relationships between objectives and principles***

These principles discussed in the example above, are also linked to other objectives as shown in Figure 3.4 below. For example, in addition to governing the achievement of 'Maximal Adaptability' as discussed above, the principle 'Simplicity' is also linked to the objectives 'Minimal Waste' and 'Continuous Innovation and Learning.' Similarly, each of the remaining principles are also linked to one or more objectives (see Figure 3.4). See Section 3.1.4 for a discussion about the linkages. Figure 3.8 provides a visual representation of the complete set of established linkages.

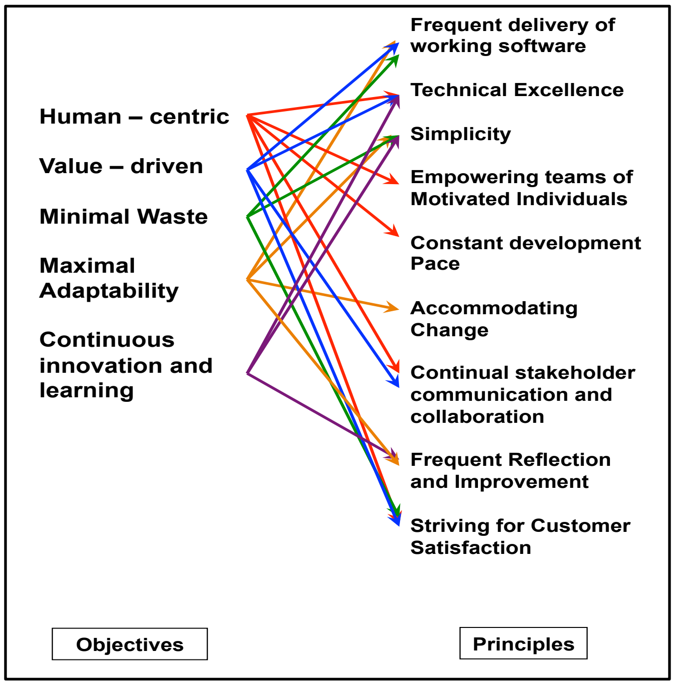

Human – centric
Value – driven
Minimal Waste
Maximal Adaptability
Continuous innovation and learning
Frequent delivery of working software
Technical Excellence
Simplicity
Empowering teams of Motivated Individuals
Constant development Pace
Accommodating Change
Continual stakeholder communication and collaboration
Frequent Reflection and Improvement
Striving for Customer Satisfaction
Objectives
Principles

***Figure 3.4. Linkages between the identified objectives and principles***

***Table 3.2. Working definitions of the identified Principles***

**Principles**

***Frequent delivery of working software***

Deliver working software frequently; iteration length – 2 to 4 weeks.

***Technical Excellence***

Provide an environment for achieving technical excellence - select the right people, right process and right practices to build working software of value to the customer.

***Simplicity***

Keep the development process simple; produce a product that displays only the necessary functionality.

***Empowering teams of Motivated Individuals***

Build teams of motivated individuals and empower them; push the decision-making process to the lower (or lowest) level.

***Constant development Pace***

Build software at a constant pace - the amount of work performed during each iteration should be constant.

***Accommodating Change***

Accommodate change with minimal impact.

***Continual stakeholder communication and collaboration***

Promote interaction among the stakeholders at regular intervals.

***Frequent Reflection and Improvement***

Re-examine the development process regularly with the intent to better understand and improve that process.

***Striving for Customer Satisfaction***

Promote customer satisfaction; provide maximum value to the customer.

The working definitions of the nine principles identified by the OPS Framework are presented in Table 3.2 (also see Appendix A). The principles presented here in this section are concise and comprehensive. Feedback gathered from agile practitioners attest to the validity and completeness of these principles (see Chapter 4).

### 3.1.3 Strategies

Strategies are implementations of the principles, i.e., they are ‘plans of action’. Agile methods provide practitioners with strategies, which are more tangible concepts, to help implement the touted principles. The OPS Framework identifies 17 such accepted strategies. Figure 3.2 depicts the set of objectives, principles, and strategies that are defined by the OPS Framework and reflective of the agile philosophy. We recognize, however, that the list of strategies shown in Figure 3.2 is not necessarily exhaustive nor complete; we do expect that it will change over time.

Let us revisit the objective Maximal Adaptability. From the discussion in section 3.1.2, we know that one of the principles that govern the achievement of the above-mentioned objective is Accommodating Change. To help implement that principle, we identify the following strategies *evolutionary requirements*, *iterative progression, incremental development*, *continuous feedback*, and *retrospection* (see Figure 3.5). Based on the working definitions (see Appendix A) and mechanics of implementation of each of these strategies, we conjecture that Accommodating Change can be implemented by using those above-mentioned strategies.

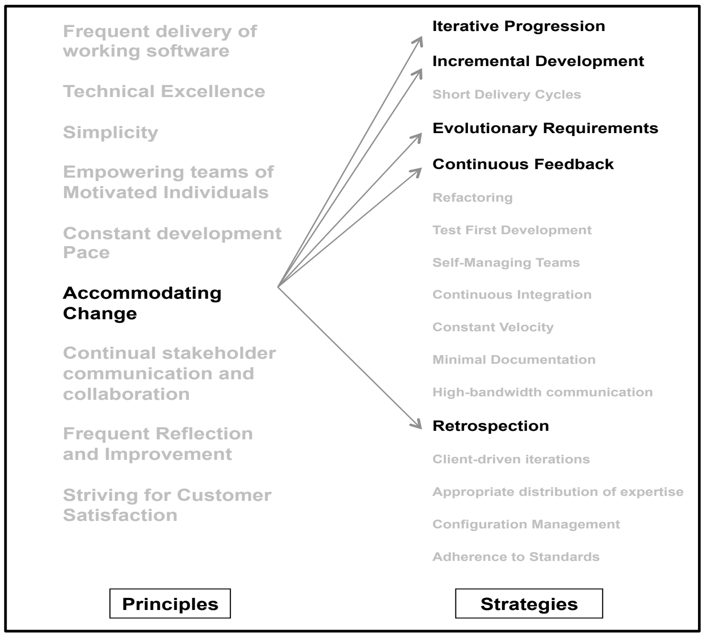


*Figure 3.5. Example relationships between Principles and Strategies*

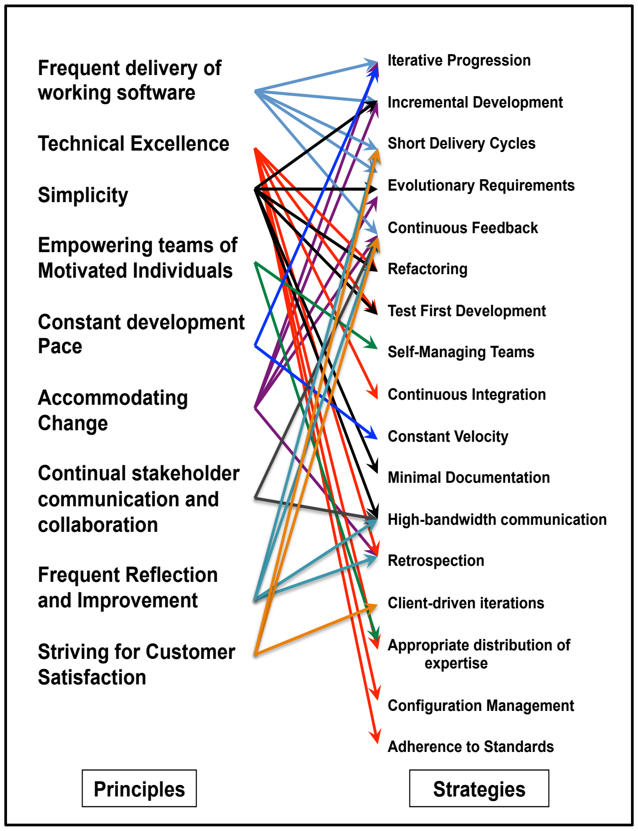


*Figure 3.6. Linkages between the principles and strategies*

Each of the strategies discussed in the example above, is also linked to other principles. Similarly, the remaining strategies are linked to one or more of the identified principles. Figure 3.6 shows the linkages between the principles and the strategies. A pictorial representation of the complete set of linkages between the objectives, principles, and strategies is presented in Figure 3.8. The working definitions of the strategies can be found in Table 3.3 below (and Appendix A).

***Table 3.3. Working definitions of the identified Strategies***

**Strategies**

***Iterative progression***

Develop the product over several iterations/cycles in sequence. Decompose the overall development lifecycle into multiple timeboxed (fixed length) release cycles and each release cycle into timeboxed iterations.

***Incremental development***

Build the product incrementally. Develop only a selected/ prioritized set of features during a release cycle.

***Short delivery cycles***

Deliver valuable software frequently.

***Evolutionary requirements***

Allow the features/ requirements to evolve over the development lifecycle.

***Continuous feedback***

Gather feedback from the customers and users on a regular basis

***Distribution of expertise***

Select the right people to complete the tasks. Ensure that the team is composed of people with the appropriate skill sets to complete the assigned tasks.

***Test-first development***

Write the unit tests first before writing code. Also, capture the customer acceptance criteria for features and stories before proceeding to the downstream development activities.

***Refactoring***

Refine the architecture, design, code, and/or other process artifacts regularly to improve the quality of that artifact by altering its internal structure while preserving its external behavior.

***Adherence to standards***

Conform to a set of standards that the team or organization has agreed to comply with. The standards adopted are guided by the culture and values of the teams and organizations. E.g. Coding standards.

***Continuous integration***

"Team members integrate their work frequently; usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible."

***Configuration management***

"Manage the evolution of the product and other artifacts, both during the initial stages of development and during all stages of maintenance."

***Minimal documentation***

Maintain just-enough documentation to satisfy the needs of the development team and the customer.

***High bandwidth communication***

Facilitate continuous communication among the stakeholders (in-person, face-to-face interactions).

***Self-managing teams***

"Allow the team members to determine, plan, and manage their day-to-day activities and duties under reduced or no supervision."

***Constant velocity***

Maintain the amount of work done during each iteration at a constant.

***Retrospection***

Re-examine the development process regularly with the intent to better understand and improve the process

***Client-driven iterations***

"The choice of features for each release comes from the client – whatever they perceive as the highest business value to them." The customers and users prioritize the features. Build only what is of value to the customers and users.

See Appendix A for the complete set of working definitions for the identified objectives, principles, and strategies. We have derived the objectives, principles, and strategies based on our understanding of the agile philosophy, our experiences and observations, existing literature, and interactions with industry experts. The substantiation results that we present in Chapter 5 indicate that the objectives, principles, and strategies, which we have identified, are consistent with practitioners' views of the agile philosophy, its values, principles, and effective strategies.

### 3.1.4 Linkages

Because the OPS Framework is intended to support an assessment process, it also defines a set of *linkages* that (a) connect objectives to supporting principles and (b) bind principles to strategies used to implement them. These linkages provide an assessment path binding the use of process principles to the achievement of stated objectives, and the use of strategies to the implementation of specific principles. These linkages are central to the assessment of adequacy, capability, and effectiveness. We have identified them using the criteria described in the following paragraph. We have currently identified and substantiated ***54 linkages*** between the objectives and the principles, and the principles and the strategies. The complete set of linkages is provided in Appendix B.

An organization or a team achieves the desired objectives by applying stated principles, which, in turn, are realized by implementing strategies. We first identify those characteristics that are crucial to and define the individual objectives, principles, and strategies. Given an understanding of the characteristics that distinguish them, we then establish a linkage between an objective and principle, or principle and strategy if we can reason that a principle would govern the achievement of an objective, or if a strategy would help implement a principle. More specifically, we would ask

- For an objective and a principle, can the defining characteristics of that objective be achieved by the application of that principle in an agile software development process?
- For a principle and a strategy, can the impact of that principle on the development process be realized by the mechanics of implementation of that strategy?

These linkages are fundamental to the measurement process. Consider the objective 'Maximal Adaptability', discussed previously. Recall that our working definition for 'Maximal Adaptability' is *maintaining (a) the flexibility to support change and (b) the freedom to choose among appropriate strategies*. We have substantiated through the work of others that one of the principles that supports the attainment of Maximal Adaptability is 'Accommodating Change'. Subsequently, as shown in Figure 3.7, there exists a linkage between the objective 'Maximal Adaptability' and the principle of 'Accommodating Change'. Evidence for the existence of this linkage can be found in [9]. To help implement this principle, we have also identified (or linked) a corresponding set of appropriate strategies (also shown in Figure 3.7). They are *evolutionary requirements*, *iterative progression*, *incremental development*, *continuous feedback*, and *retrospection*. The linkages between Accommodating Change and the five strategies given above have been substantiated using the sources [16], [8], [17], and [4].

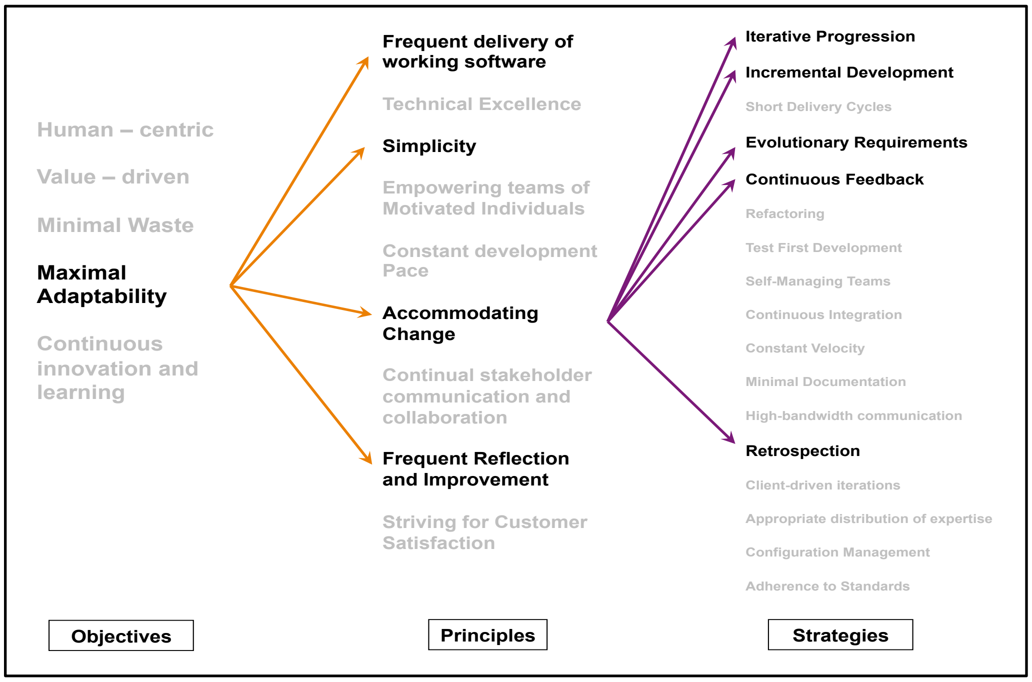


*Figure 3.7. Example linkages in the OPS Framework*

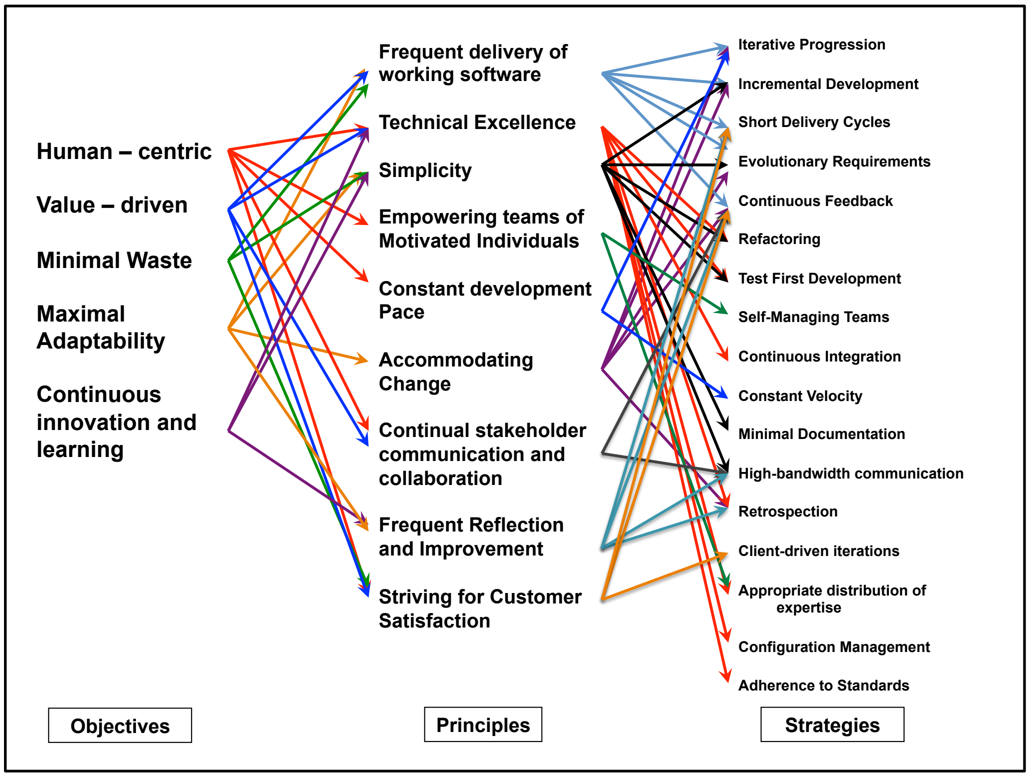


*Figure 3.8. Established Linkages in the OPS Framework.*

Although linkages between the objectives, principles, and strategies have not been explicitly enunciated in the existing literature, our research efforts have enabled us to definitively identify, document, and substantiate their existence [18] through independent sources (see Appendix B). Figure 3.8 shows the complete set of established linkages between the objectives and principles, and the principles and strategies.

### 3.1.5 Indicators

As discussed previously, to assess (i) the capability of an organization to support the implementation of an agile method and (ii) the effectiveness of the method itself, we employ a bottom-up traversal of the established linkages. We define strategies at the lowest level of the OPS framework; our assessment of capability and effectiveness is initiated at this level. To determine if the organization has the supporting environment to effectively implement a strategy, or if the strategy has produced the intended results, we measure properties that attest to the existence or use of that strategy. These properties are observable characteristics of the people, process, project, product, and environment, and are specific to each strategy. Because of this intentionally-formed and undeniable relationship, *we consider each strategy and property pair to be an indicator.* Indicators are directly measurable and are tailored to assess the strategies.

***Table 3.4. Properties associated with Evolutionary Requirements***

***Property: Minimal Big Requirements Up Front (BRUF) and Big Design Up Front (BDUF) (Process)***

Is it expected

- That only the high level features be identified upfront?
- An evolutionary approach to architecting the system be followed as opposed to creating the architecture upfront?

***Property: Feature decomposition (Process)***

Is it expected that a mechanism for decomposing the selected features to be developed during the current release cycle into stories be defined?

***Property: Just-In-Time Refinement (Process)***

Is it expected that the requirements be determined and refined just-in-time?

***Property: Requirements Reprioritization (Process Artifacts)***

Are the features reprioritized as and when new features are identified?

***Property: Customer satisfaction (Process Artifacts)***

To what extent are the changes requested by the customers accommodated?

The first step in defining the indicators is to identify the observable properties of the people, process, project, and product associated with each strategy. Consider the strategy ***Evolutionary Requirements***. Our working definition for this strategy is "Allow the features/ requirements to evolve over the development lifecycle" (see Appendix A). By analyzing the mechanics of implementation for Evolutionary Requirements, we identify the observable properties given in Table 3.4. Associated with each property, is a set of questions that help measure the extent to which the property is supported or achieved.

The indicators in this example are the following:

- (Evolutionary Requirements, Minimal BRUF and BDUF)
- (Evolutionary Requirements, Feature Decomposition)
- (Evolutionary Requirements, JIT Refinement)
- (Evolutionary Requirements, Requirements Reprioritization)
- (Evolutionary Requirements, Customer Satisfaction)

These questions are separate measures of the indicators, i.e., the (strategy, property) pairs. For each question, the corresponding measurement scale can be subjective, objective, binary, or range values. In Table 3.4, the measurement scales used are binary and subjective. Appendix C provides the set of identified properties associated with the strategies and the complete set of questions used to measure the indicators.

Table 3.5 lists the number of questions used to measure each indicator (i.e. strategy, property pair) from both a capability and effectiveness perspective. Currently, the OPS Framework identifies (a) ***80 indicators*** tied to the ***17 strategies*** and (b) ***150 questions*** to measure those ***80 indicators***.

***Table 3.5. Total number of indicators and questions identified by the OPS Framework***

| **Strategy** | **Capability** | | **Effectiveness** | |
|---|---|---|---|---|
| | **No. of Indicators** | **No. of Questions** | **No. of Indicators** | **No. of Questions** |
| Iterative Progression | 3 | 3 | 3 | 6 |
| Incremental Development | 3 | 4 | 2 | 5 |
| Short Delivery Cycles | 1 | 1 | 3 | 4 |
| Evolutionary Requirements | 3 | 4 | 3 | 4 |
| Continuous Feedback | 2 | 2 | 2 | 2 |
| Distribution of Expertise | 1 | 3 | 1 | 5 |
| Test-first development | 3 | 4 | 3 | 4 |
| Refactoring | 3 | 8 | 2 | 5 |

| | | | | |
|---|---|---|---|---|
| Adherence to standards | 5 | 6 | 2 | 3 |
| Continuous Integration | 4 | 10 | 3 | 10 |
| Configuration Management | 2 | 6 | 1 | 1 |
| Minimal Documentation | 3 | 3 | 1 | 4 |
| High bandwidth Communication | 4 | 9 | 3 | 13 |
| Self-Managing Teams | 3 | 3 | 3 | 7 |
| Client-driven Iterations | 1 | 2 | 3 | 3 |
| Retrospection | 2 | 2 | 2 | 4 |
| **Total** | **43** | **70** | **37** | **80** |

### 3.1.6 Summary

The OPS Framework provides the foundation for the assessment of the ‘goodness’ of agile methods. The objectives, principles, strategies, linkages, and indicators are the core components of the Framework. We assess the adequacy of Basis and Instance methods by employing a top-down traversal of the linkages from the objectives to the principles, and from the principles to the strategies. To assess capability and effectiveness, a complementary bottom-up traversal of the linkages is initiated. Unlike adequacy, capability and effectiveness are examined from an organizational perspective. Hence, we can assess capability and effectiveness only for the Instance methods. Indicators, which are (strategy, property) pairs, are central to the assessment of capability and effectiveness. The difference in the two assessment perspectives lies in the types of indicators used. We examine people, process, project, and environment indicators to assess capability. For effectiveness, we evaluate indicators of the product and process artifacts. We measure the indicators and the values are propagated up to the higher levels of the hierarchy. Questions associated with each indicator help measure that indicator. The next sections discuss the approach to assessing adequacy, capability, and effectiveness.

## *3.2 Assessment Methodology*

Using the OPS Framework described in Section 3.1, we assess the 'goodness' of an agile method by (a) assessing its adequacy, (b) the capability of the organization to provide the supporting environment to implement the method, and (c) the effectiveness of that method. In this section, we describe our assessment methodology.

### 3.2.1 Assessing Adequacy

Recall that adequacy is defined as *the sufficiency of an agile method to meet its stated objectives*. Assessing adequacy is *independent* of any organizational characteristics. More specifically, we can assess the adequacy of Basis methods such as eXtreme Programming (XP) [19], Lean [8], Crystal [20], Feature Driven Development (FDD) [21], or any tailored instances thereof, with respect to the agile objectives and principles each espouses. To assess adequacy, as shown in Figure 3.9, we follow the linkages in a *top-down* fashion from the objectives to the principles, and finally to the strategies.

Given an agile method, we examine its structure and composition to determine its adequacy. That is, guided by the OPS Framework, for agile method X we ask the following questions:

1. Does Method X tout objectives that are consistent with those stated by the OPS Framework?

   Consistency is confirmed if the set of objectives touted by Method X is equal to or a subset of those expressed by the Framework. We accept subsets because organizations often tailor their agile development approach to reflect cultural distinctiveness and to emphasize their business goals and values. We would question, however, any objective highlighted by Method X that is not one of those enunciated by the OPS Framework.

2. Does Method X state principles that support the achievement of those touted objectives?

   Using the set of objectives articulated by Method X, we first identify the same set embodied within the OPS Framework, and then follow the linkages from those objectives to the corresponding set of supporting principles. This set of principles is precisely those that must be enunciated by Method X to support the realization of its publicized objectives. An inconsistency between the two sets of principles indicates a potential deficiency in the ability of Method X to achieve its stated objectives.

3. Does Method X express the appropriate set of strategies that implement its stated principles?

   We use the principles enunciated by Method X to identify a corresponding set of principles within the OPS Framework, and then follow the Framework linkages from its principles to a related set of strategies defined within the OPS Framework. This identified set of strategies support the implementation of the principles to which they are connected. Hence, for Method X to sufficiently implement the principles it enunciates, its touted set of strategies must correspond to the set of OPS Framework strategies as determined above. An inconsistency between the two sets of strategies indicates a potential deficiency in the ability of Method X to implement its stated principles.

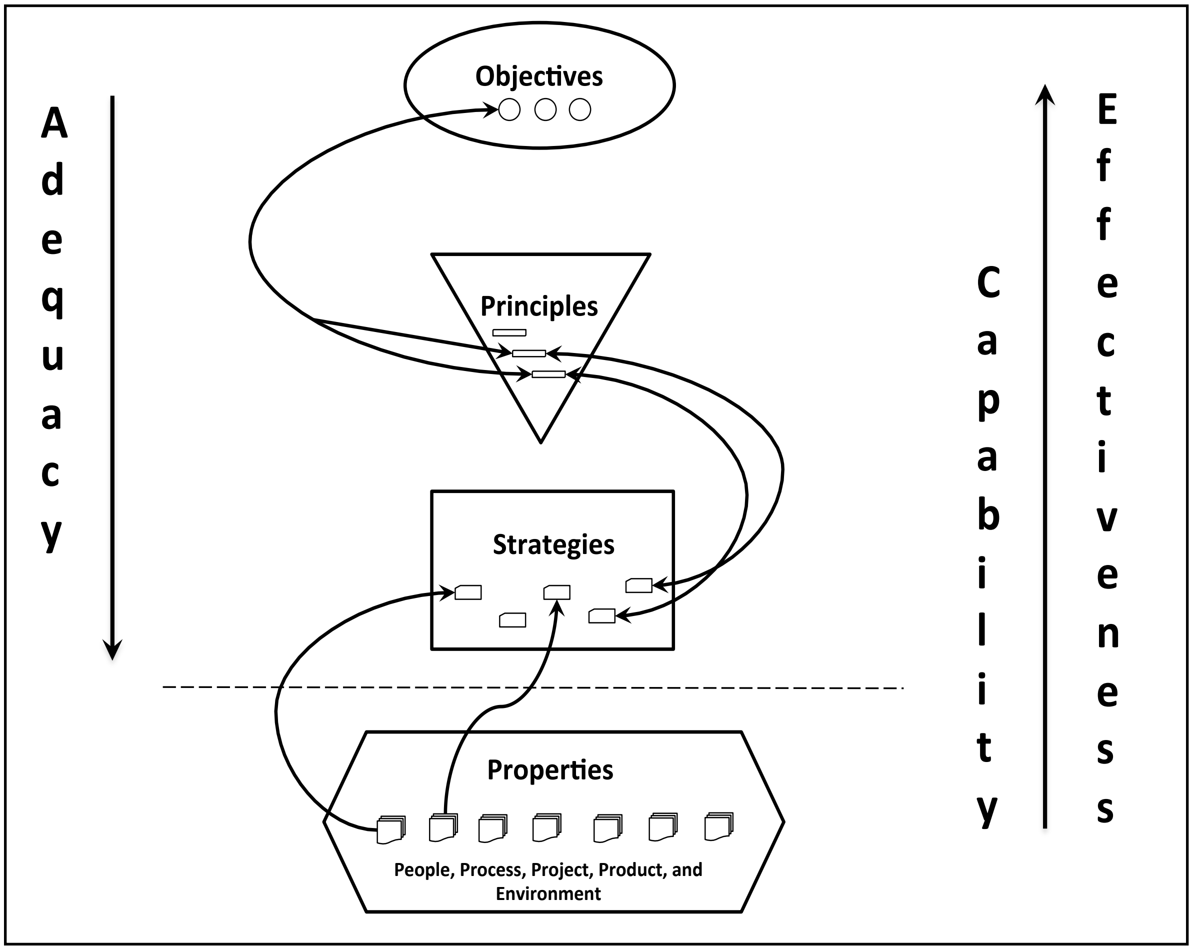


*Figure 3.9. Assessing Adequacy, Capability, and Effectiveness*

If necessary principles and strategies are missing, then adequacy of the method is suspect.

Clearly, the procedure outlined above assumes that the OPS Framework is, necessary and sufficient for such comparisons. As described previously, our effort to consolidate the works of many, and to substantiate the linkages prescribed within the OPS Framework, provide evidence that a 3-Step examination process like the above is justified. Nonetheless, we will also be the first to state that sets of objectives, principles, strategies, and linkages are not intended to be closed set. As such, we continually revisit the composition of each. Following the process outlined above, we have assessed the adequacy of two Basis and one Instance methods. We present the results in Chapter 4.

### 3.2.2 Assessing Capability and Effectiveness

Unlike adequacy, both capability and effectiveness are assessed from an organizational perspective. That is, capability and effectiveness are assessed for agile methods that have been adopted by organizations. *Capability* assessment examines the extent to which an organization has the means to implement its adopted method. *Effectiveness*, on the other hand, is measured relative to how well the adopted method actually achieves its stated objectives. The two assessment approaches are computed independently using *both* a *top-down* and *bottom-up* traversal of the linkages (see Figure 3.9). The top-down traversal is necessary because method adequacy directly impacts capability and effectiveness. Bottom-up traversal, however, is predicated on the identification of people, process, project, and product properties that reflect the presence, implementation and use of an associated strategy. More specifically, for bottom-up assessment we examine the environment, people, process, project, or product for properties that definitively attest to the presence and usage of a specific strategy (see Figure 3.9). In turn, this information is used to determine adherence to stated principles, and ultimately, the achievement of a stated objective. We refer to each strategy, property pair as an *indicator*. Indicators are the fifth, and final, major component of the OPS Framework. *The difference between assessing capability or effectiveness, however, lies primarily in the type of indicators used*. That is, we use environment, people, process, and project indicators to assess capability; we use process artifacts and product indicators to assess effectiveness. As shown in Figure 3.9, by following the linkages upward from the indicators, we can infer the use of proper principles and the achievement of desired objectives.

### *3.2.2.1 Assessing Capability*

We define the capability of an organization as its ability to provide the supporting environment conducive to the implementation of an agile method. In assessing the capability of an organization, we are concerned with the characteristics of its *internal environment*. The internal environment of an organization is primarily composed of its resources and competencies. More specifically, in an organization, the characteristics of its people, the process that it adopts, its environment, and its projects are reflective of the characteristics of its internal environment. Hence, we use observable properties of the environment, people, process, and project in our assessment of capability. For example, the presence of open physical environments in an organization is *indicative* of the organization's capability to foster face-to-face stakeholder communication and collaboration at any given time.

We identify observable properties of the people, process, project and the environment for every strategy by asking the following questions:

1. What special skills or knowledge do the people involved in the project need to successfully adopt and implement the strategy?
2. What characteristics of the process and/or the environment extend support for the implementation of the strategy?
3. Are there any project specific characteristics that support or impinge on the effective realization of the strategy?

Consider the agile strategy *Retrospection* identified by the OPS Framework. Our working definition for retrospection is "Re-examine the development process regularly with the intent to better understand and improve the process" (see Appendix A). Retrospection is also one of the tenets of the agile philosophy. Retrospective meetings are usually scheduled at the end of each iteration and release cycle. Team reflect on the 'things that went well', 'things that didn't go so well' and 'how can we improve?' The team then sets retrospective goals that will be achieved over the next iterations. Asking the questions mentioned above with respect to retrospection provides us with the following properties:

***Property: Support for retrospection (Process property)***
Is retrospection an expected activity?

***Property: Tool support (Environment property)***
Are tools available for recording the outcomes of the retrospective meetings?

As discussed previously, a set of questions is associated with each indicator to measure that indicator. The complete set of the objectives, principles, strategies, properties, and questions with the linkages and associations among them, form a multi-level hierarchical structure that we refer to as the *indicator hierarchy*.

Recall that to assess capability, we identify properties of the people, process, project, and environment. Hence, the indicator hierarchy for assessing capability includes the above-mentioned properties. Guided by the indicator hierarchy, we initiate the capability assessment by seeking answers to the questions linked to each property. Those answers are recorded as numeric values using a pre-defined scale. A property value is then computed as a weighted average of its linked scores. In turn, values for each strategy are computed as an average of its associated property values. Following the same weighted average process, values are propagated from the strategies level, through the principles level, and up to the objectives level. For the computations, we have assigned equal weights to all the elements. Using this process, we have assessed the capability of four agile teams in Company Z (see Chapter 6).

### *3.2.2.2 Assessing Effectiveness*

An agile method is judged effective if it produces "*working software*" that is a *quality product*, *on time*, *within budget*, and *of value to the customers*. Assessing the effectiveness of the agile method involves the identification of properties of process artifacts and the product, which focus on the system produced and its value to the stakeholders. For example, agile teams strive to maintain constant velocity (the empirical observation of the work done by a team during an iteration [22]). Velocity is reflective of the efficiency and effectiveness of the team. Indicators of constant velocity can be obtained by studying Velocity, Burn-Up and Burn-Down charts. We ask the following questions to determine the observable properties of process artifacts and the product:

1. Are there any process artifacts generated when implementing the strategy? If so, what are the criteria for the verification and validation of these artifacts?
2. What are the criteria to determine the successful implementation of the strategy? What properties are observed in the final product that is a direct result of the successful implementation of the strategy under consideration?

Let us revisit the agile strategy Retrospection discussed previously. Asking the questions mentioned above to determine the properties of the product and process artifacts with respect to retrospection provides us with the following:

***Property: Team Scheduling (Process Artifacts)***

Is time allocated for retrospection at the end of each iteration and release cycle?

***Property: Process Outcomes (Process Artifacts)***

To what extent were the following identified during the retrospection meeting

1. Approaches that worked well during the iteration or the release cycle and hence should be used in the future iterations
2. Approaches that did not yield the expected results and hence should be discontinued
3. New approaches that may better suit the team's needs
4. Impediments
5. Success factors
6. Retrospective goals

***Property: Refining the process based on previous retrospective activities (Process Artifacts)***

To what extent was the process during the current iteration or release cycle refined based on the observations from the previous retrospective meetings?

Similar to the capability assessment process, we construct an indicator hierarchy for assessing effectiveness. This hierarchy includes properties of the product and process artifacts. The assessment is initiated at the strategies level. The answers to the questions used to measure the indicators are recorded on a pre-defined numeric scale. For each strategy, a weighted average of the scores of its associated properties is computed. Following the same process, these weighted average scores are propagated to the higher levels of the hierarchy. We have cursorily examined the effectiveness of the methods adopted by four agile teams at Company Z. We have followed the systematic process outlined in this section. The results are provided in Chapter 6.

### 3.2.3 Summary

In this research we present a systematic and comprehensive approach to assessing the 'goodness' of agile methods. 'Goodness' is determined from the three perspectives of adequacy, capability, and effectiveness. In assessing adequacy, we examine the composition of an agile method to determine if the method supports necessary principles and strategies to achieve its stated objectives. A top-down traversal of the linkages from the objectives to the principles, and from the principles to the strategies is initiated to assess adequacy. To assess capability and effectiveness, we traverse the linkages in a bottom-up fashion. The bottom-up traversal is initiated after assessing the adequacy of the method under consideration. Capability examines the extent to which an organization or a team can support the implementation of its adopted agile method(s). Effectiveness defines the extent to which that method under consideration produced expected results. To assess capability and effectiveness, we identify observable characteristics of the people, process, project, product, and environment. These characteristics are associated with the strategies. Each (strategy, property) pair is an indicator. Each indicator is associated with a set of questions that are used to measure that indicator. Scores assigned to the answers to these questions are aggregated and propagated to the higher levels of the indicator hierarchies. We recognize that we may not observe properties of all the aspects of people, process, project, product, and environment for all strategies. The working set of properties is provided in Appendix C. Our approach to substantiating the OPS Framework and the assessment methodology, results, and observations are described in the next chapters.

## REFERENCES

[1] D. Ulrich and N. Smallwood (2004), "Capitalizing on Capabilities," *Harvard Business Review,* vol. 82, pp. 119-127.

[2] J. D. Arthur and R. E. Nance (1991), "A Framework for Assessing the Adequacy and Effectiveness of Software Development Methodologies," Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

[3] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn*, et al.* (2001), *Manifesto for Agile Software Development*, http://www.agilemanifesto.org

[4] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta (2002), *Agile Software Development Methods: Review and Analysis*. Finland: VTT Publications.

[5] J. Shore and S. Warden (2008), *The Art of Agile Development*. Sebastopol, CA: O'Reilly Media, Inc.

[6] G. Smith and A. Sidky (2009), *Becoming Agile in an Imperfect World*. Greenwich, CT: Manning Publications Co.

[7] J. Highsmith (2002), *Agile Software Development Ecosystems*. Addison-Wesley.

[8] M. Poppendieck and T. Poppendieck (2003), *Lean Software Development: An Agile Toolkit*. Addison Wesley.

[9] A. Koch (2004), *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House Publishers.

[10] A. Qumer, B. Henderson-Sellers, and T. McBride (2007), "Agile Adoption and Improvement Model," in *European, Mediterranean & Middle Eastern Conference on Information Systems (EMCIS)*, Polytechnic University of Valencia, Spain.

[11] A. Qumer and B. Henderson-Sellers (2008), "An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering," *Information and Software Technology,* vol. 50, pp. 280-295.

[12] C. Schwaber (2007), "The Truth About Agile Proceses - Frank Answers to Frequently Asked Questions," Forrester Research.

[13] J. Patton (2008), "Ambiguous Business Value Harms Software Products," *IEEE Software,* vol. 25, pp. 50-51.

[14] J. Johnson (2002), "The Cost of Big Requirements up Front (Bruf)," presented at the XP (eXtreme Programming) 2002 Conference, Alghero, Sardinia.

[15] K. Beck (2003), *Test-Driven Development: By Example*. Addison-Wesley.

[16] L. Cao and B. Ramesh (2008), Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*.

[17] T. Dingsøyr and G. K. Hanssen (2003), "Extending Agile Methods: Postmortem Reviews as Extended Feedback," in *Advances in Learning Software Organizations*. vol. 2640, S. Henninger and F. Maurer, Eds., ed: Springer Berlin / Heidelberg, pp. 4-12.

[18] S. Soundararajan (2011), "A Methodology for Assessing Agile Software Development Approaches," *Computing Research Repository,* vol. abs/1108.0427.

[19] K. Beck and C. Andres (2004), *Extreme Programming Explained: Embrace Change*, 2 ed.: Addison-Wesley Professional.

[20] A. Cockburn (2004), *Crystal Clear: A Human-Powered Methodology for Small Teams*: Addison-Wesley Professional.

[21] S. R. Palmer and J. M. Felsing (2002), *A Practical Guide to Feature Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR.

[22] D. Nicolette (2009), *Agile Metrics*, http://www.slideshare.net/rsrivastava91/agile-metrics-v6

# 4 Substantiating the Components of the OPS Framework

The OPS Framework guides our assessment process. Our goal is to substantiate both the components of the OPS Framework, and our process for assessing adequacy, capability, and effectiveness. In this chapter, we describe our efforts to substantiate the components of the OPS Framework. We have administered three surveys to gather feedback from the agile community about (1) the core components of the Framework, and (b) the relevance and viability of the Framework and the assessment methodology. One of the surveys was available online and the other two were paper-based and were administered at the Agile 2012 Conference and at Company Z respectively. The results from these surveys have helped refine our work.

## *4.1 Online Survey*

Previously, the assessment Framework described in this dissertation was entitled the Objectives, Principles, and Practices (OPP) Framework. It identified objectives of the agile philosophy, principles that govern the achievement of those objectives, and practices that were implementations of the principles. Definitive linkages between the above-mentioned components were also established. For more information about the OPP Framework, see [1-3]. As a first step towards substantiating the components of the OPP Framework, we gathered feedback from agile practitioners about the identified objectives, principles, and practices. The OPP Framework identified five objectives of the agile philosophy, nine principles that govern the achievement of those principles, and 27 practices. *The set of objectives, and principles identified by the OPP Framework are the same as those shown in Figure 3.2.* To substantiate these foundational pieces, we designed a survey instrument (implemented online) to gather practitioners' opinions about the identified objectives, principles, and practices. This survey was administered to determine the extent to which that the identified objectives, principles, and practices are consistent with practitioners' views of the agile philosophy and the development paradigm. We have evolved the OPS Framework based on practitioners' feedback.

***Demographics***

The survey included initial questions to gather demographic information about the respondents, e.g. the number of years of experience they have working in agile teams, their level of expertise in Agile Software Development, the roles which they held within teams, etc.

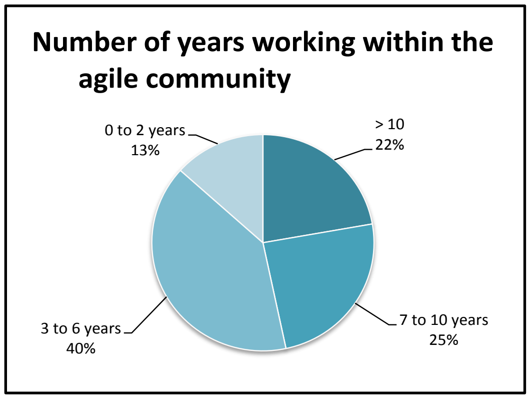


***Figure 4.1. Survey Responses - Experience***

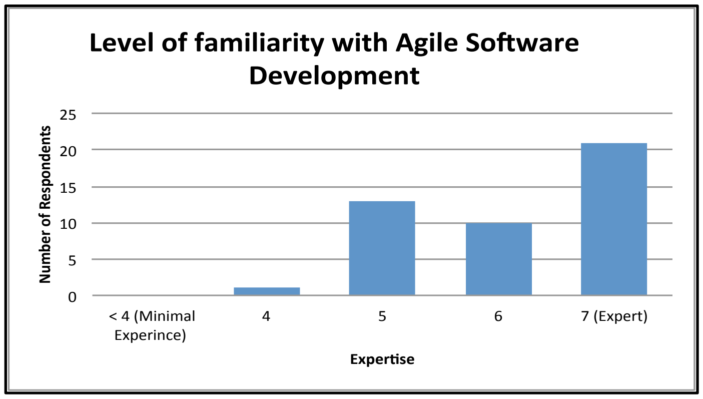


***Figure 4.2. Survey Responses - Level of Familiarity***

We sent invitations to a host of agile practitioners, requesting that they complete the survey. Thus far, 45 practitioners have completed the survey. Demographically, 29 of the survey respondents have been developers, 21 of them have coached agile teams, and 21 of them have managed projects in the capacity of a scrum master. Of these practitioners, 21 have been working within the agile community for 7 or more years. They account for 47% of the respondents. 40% of the survey respondents have 3 to 6 years experience working in agile environments (see Figure 4.1).

We also asked practitioners to rate their level of familiarity with Agile Software Development (see Figure 4.2). The scale was based on a 1-7 rating with level 1 indicating minimal experience and level 7 implying that the person is an expert in the Agile Software Development paradigm. 21 Practitioners (47% of the survey respondents) have stated that they consider themselves to be Experts.

***Components of the OPP Framework***

In designing the survey, our goal has been to determine if the objectives, principles, and practices identified by the OPP Framework are consistent with practitioners' views of the agile philosophy, its values, principles, and effective practices.

The following questions were asked of each participant:

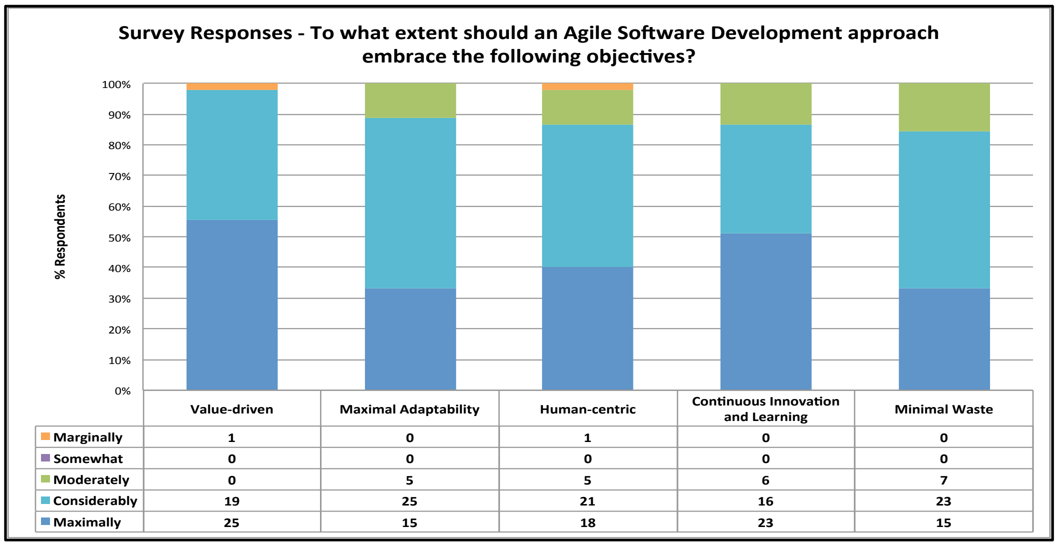
1. To what extent (maximally, considerably, moderately, somewhat, or marginally) should an Agile Software Development approach embrace the objectives identified by the OPP Framework?

2. To what extent (maximally, considerably, moderately, somewhat, or marginally) should an Agile Software Development process promote the principles suggested by the OPP Framework?

3. How important (very important, considerably, moderately, somewhat, or marginally important) are the identified practices in supporting an effective Agile Software Development process?

***Objectives***

As discussed previously, the set of objectives identified by the OPP Framework are: human-centric, value-driven, minimal waste, maximal adaptability, and continuous innovation and learning. More than 85% of the survey respondents state that an Agile Software Development approach should "***maximally***" or "***considerably***" embrace each of the objectives identified by the OPP Framework (see Figure 4.3).



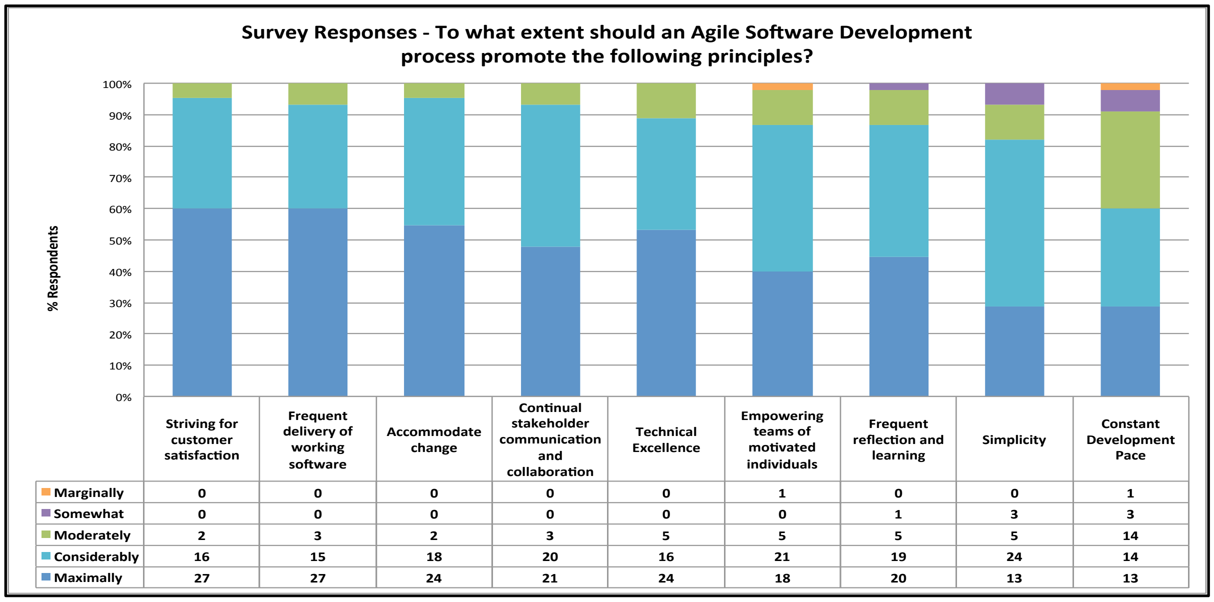
| | Value-driven | Maximal Adaptability | Human-centric | Continuous Innovation and Learning | Minimal Waste |
|---|---|---|---|---|---|
| 🟧 Marginally | 1 | 0 | 1 | 0 | 0 |
| 🟪 Somewhat | 0 | 0 | 0 | 0 | 0 |
| 🟩 Moderately | 0 | 5 | 5 | 6 | 7 |
| 🟦 Considerably | 19 | 25 | 21 | 16 | 23 |
| 🟦 Maximally | 25 | 15 | 18 | 23 | 15 |

***Figure 4.3. Survey Responses - Objectives***

***Principles***

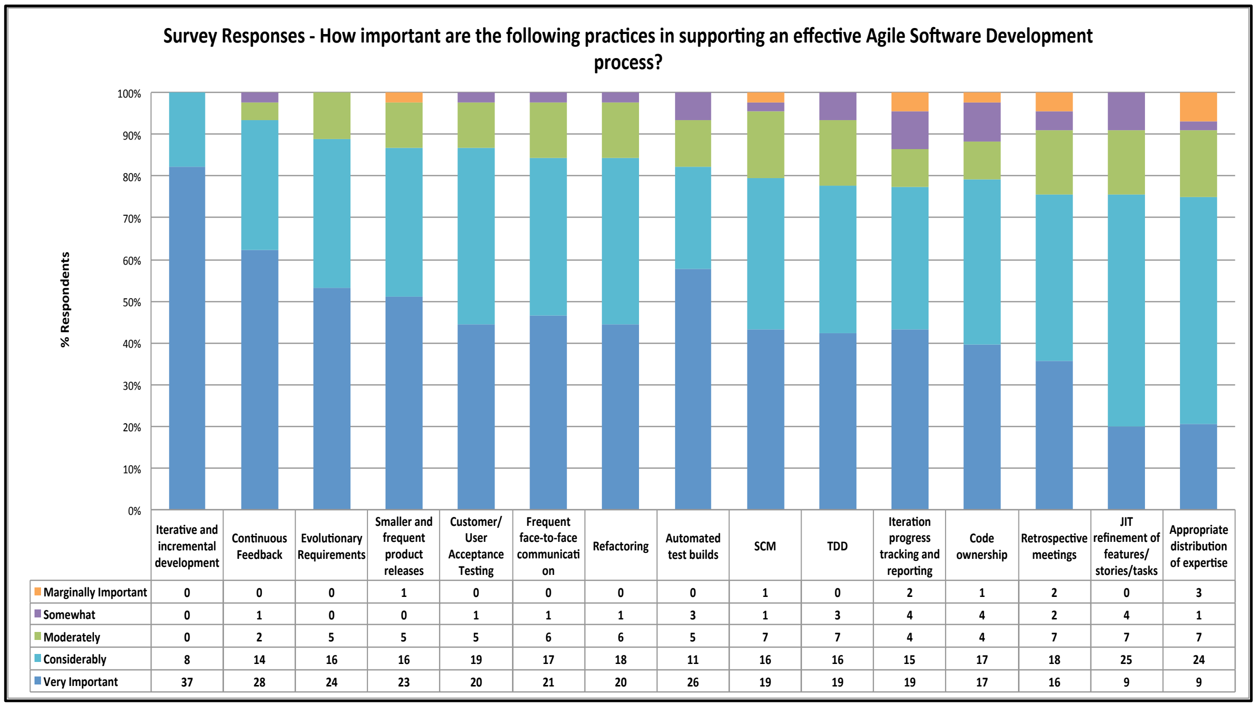
As shown in Figure 4.4, with the exception of Constant Development Pace, 80% of the respondents also state that an Agile Software Development process should "***maximally***" or "***considerably***" promote each of the principles identified by the OPP Framework (see Figure 3.2).

*Since the Framework, in its current (OPS) state, identifies the same set of objectives and principles as its previous version (OPP), the results from the survey with respect to the objectives and principles (see Figures 4.3 and 4.4) are equally applicable.*



| | Striving for customer satisfaction | Frequent delivery of working software | Accommodate change | Continual stakeholder communication and collaboration | Technical Excellence | Empowering teams of motivated individuals | Frequent reflection and learning | Simplicity | Constant Development Pace |
|---|---|---|---|---|---|---|---|---|---|
| Marginally | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Somewhat | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 3 |
| Moderately | 2 | 3 | 2 | 3 | 5 | 5 | 5 | 5 | 14 |
| Considerably | 16 | 15 | 18 | 20 | 16 | 21 | 19 | 24 | 14 |
| Maximally | 27 | 27 | 24 | 21 | 24 | 18 | 20 | 13 | 13 |

*Figure 4.4. Survey Responses - Principles*



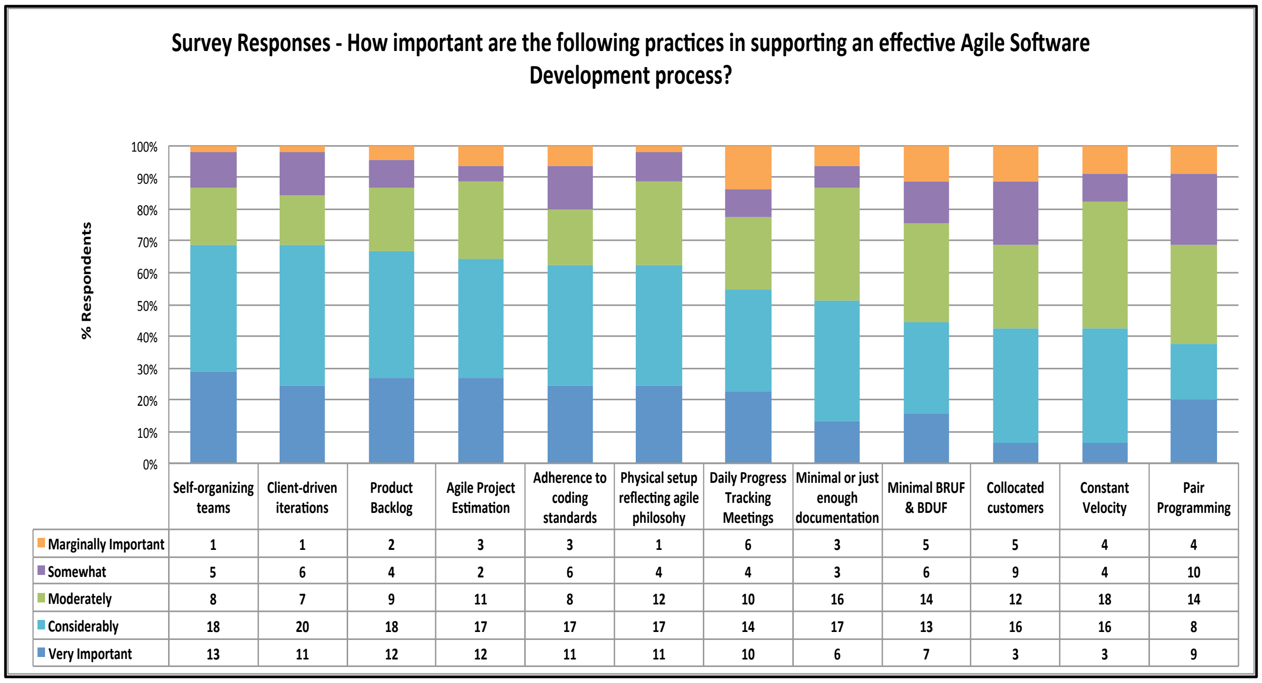
| | Iterative and incremental development | Continuous Feedback | Evolutionary Requirements | Smaller and frequent product releases | Customer/ User Acceptance Testing | Frequent face-to-face communication | Refactoring | Automated test builds | SCM | TDD | Iteration progress tracking and reporting | Code ownership | Retrospective meetings | JIT refinement of features/ stories/tasks | Appropriate distribution of expertise |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Marginally Important | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 1 | 2 | 0 | 3 |
| Somewhat | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 3 | 1 | 3 | 4 | 4 | 2 | 4 | 1 |
| Moderately | 0 | 2 | 5 | 5 | 5 | 6 | 6 | 5 | 7 | 7 | 4 | 4 | 7 | 7 | 7 |
| Considerably | 8 | 14 | 16 | 16 | 19 | 17 | 18 | 11 | 16 | 16 | 15 | 17 | 18 | 25 | 24 |
| Very Important | 37 | 28 | 24 | 23 | 20 | 21 | 20 | 26 | 19 | 19 | 19 | 17 | 16 | 9 | 9 |

*Figure 4.5. Survey Responses - Practices Viewed As Very Important or Considerably Important*

## *Practices*

With respect to the identified practices, however, the practitioners state that only a subset (15/27) of the identified practices are important to Agile Software Development, and that the remaining ones are less important. More specifically, as shown in Figure 4.5, 75% or more of the survey respondents state that only 15 practices are "***very***" or "***considerably***" ***important***.

Figure 4.6 shows the remaining 12 practices that are deemed relatively unimportant by the survey respondents.



| | Self-organizing teams | Client-driven iterations | Product Backlog | Agile Project Estimation | Adherence to coding standards | Physical setup reflecting agile philosohy | Daily Progress Tracking Meetings | Minimal or just enough documentation | Minimal BRUF & BDUF | Collocated customers | Constant Velocity | Pair Programming |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Marginally Important | 1 | 1 | 2 | 3 | 3 | 1 | 6 | 3 | 5 | 5 | 4 | 4 |
| Somewhat | 5 | 6 | 4 | 2 | 6 | 4 | 4 | 3 | 6 | 9 | 4 | 10 |
| Moderately | 8 | 7 | 9 | 11 | 8 | 12 | 10 | 16 | 14 | 12 | 18 | 14 |
| Considerably | 18 | 20 | 18 | 17 | 17 | 17 | 14 | 17 | 13 | 16 | 16 | 8 |
| Very Important | 13 | 11 | 12 | 12 | 11 | 11 | 10 | 6 | 7 | 3 | 3 | 9 |

***Figure 4.6. Survey Responses - Practices that are considered Less Important***

Agile methods focus on the product, i.e., "working software is the primary measure of progress" and working software is valued more than comprehensive documentation. More specifically, agile methods are product-oriented or product-centric. Also, they are results-driven. That is, they are designed to produce software of value to the customers frequently and in small increments. Analyzing the mechanics of implementation of each of the practices in Figure 4.5 except 'Appropriate Distribution of Expertise', it is clear that every practice shown in that figure either helps produce or refine a product increment or certain process artifacts. In short, they are product-centric. All of these practices produce or act on tangible process artifacts and the product. On the other hand, the practices that are considered less important (see Figure 4.6) have been devised to improve the process or to make the process consistent with the agile philosophy.  That is, they are process-centric. The agile philosophy states that individuals and interactions are valued more than processes and tools. The survey responses reflect that practitioners consider the practices that are process-centric to be less important than those that focus on the product.

The assessment approach presented in this dissertation is comprehensive and focuses on the people, process, project, and product characteristics. Although the 12 practices given in Figure 4.6 are considered less important, it is our opinion that practices such as Self-organizing teams, Agile Project estimation, Adherence to coding standards, Physical setup reflecting the Agile philosophy, and Daily progress tracking meetings are desirable.

***Practices to Strategies***

From Figures 4.5 and 4.6, it is clear that practitioners consider only some of the agile practices to be important in supporting an effective Agile Software Development process and the others relatively unimportant. We hypothesize that some of the differences and variations is due to the fact that multiple practices often overlap relative to implementing the same principle. We also realize that the feedback obtained was not indicative of the importance of particular practices, but of higher-level process concepts, i.e. strategies. That is, the identified practices are specific instances of development strategies. Guided by this insight and the results, we have evolved the Framework to now present a set of ***strategies***. The strategies are defined as 'plans of action' that implement the touted principles (see Chapter 3). Figure 3.2 shows the set of strategies identified by the OPS Framework.

***Table 4.1. Abstractions of Practices to Strategies***

| Practices | Strategies |
|---|---|
| Minimal BRUF and BDUF<br>Product Backlog<br>Agile project estimation | **Evolutionary Requirements** |
| Frequent face-to-face communication<br>Collocated customers<br>Open Physical Environment | **High-bandwidth communication** |
| Code ownership<br>Pair Programming | **Adherence to Standards** |
| Automated test builds<br>Daily builds<br>Multiple integrations per day | **Continuous Integration** |
| Customer and User Acceptance Testing<br>Collocated customers<br>Identifying and refining features | **Continuous Feedback** |

As discussed in the previous paragraph, we have aggregated the practices to form higher levels of abstractions, which we call strategies. Table 4.1 shows some of the abstractions of practices to strategies that we have defined. Consider the strategy 'Evolutionary Requirements.' The working definition of this strategy is 'Allow the features/ requirements to evolve over the development lifecycle' (see Appendix A). In Table 4.1, we see that the practices 'Minimal BRUF and BDUF', 'Product Backlog' and 'Agile Project Estimation' have been aggregated to form the strategy 'Evolutionary Requirements.' Analyzing the mechanics of implementation of the strategy under consideration and the above-mentioned practices, we see that the three practices are instances of 'Evolutionary Requirements.' For example, specifying the minimum set of requirements, and a "bare bones" architecture upfront is a specific instance of implementing 'Evolutionary Requirements.' Similarly, we have identified other aggregations of practices to form the strategies (see Table 4.1). The complete set of strategies is shown in Figure 3.2 and the working definitions are provided in Appendix A.

## 4.2 Surveys at the Agile 2012 Conference and at Company Z

In designing and implementing the online survey (see Section 4.1), our intent has been to gather practitioner feedback about the identified objectives, principles, and practices. As discussed above, the results from the survey indicated the presence of strategies that are higher-level abstractions. The Framework has been refined to reflect those strategies. In addition to soliciting responses about the core components of the Framework, we recognized the need for gathering feedback about the ***relevance*** and ***viability*** of the OPS Framework and assessment approach. More specifically, using this second set of surveys, our intent has been to gauge practitioner responses to the following questions:

(a) Is there a need for a *comprehensive agile assessment approach*?

(b) Does the OPS Framework and assessment methodology *provide guidance* to organizations to define or refine their adopted agile methods?

(c) Is the OPS-based assessment approach *viable*?

Hence, we administered two surveys – one at the Agile 2012 conference and the other at Company Z. The latter survey was an expanded version of the former. To introduce our work to the survey respondents, in both cases, presentations about the OPS Framework and the assessment approach were delivered. In this sub-section, we discuss the demographics and survey results for each of the three components identified above. The survey instruments are provided in Appendix D.

***Demographics***

The author of this dissertation had the opportunity to present her work at the Agile 2012 Conference. We recognized that the conference would be an ideal platform to gather practitioner feedback. Hence, we requested the practitioners who attended the session to complete the survey. The second survey was administered at Company Z after we introduced our work to the personnel of that organization. 23 practitioners completed the survey at the conference and 7 others at Company Z. Among the combined 30 respondents, 16 of them have coached agile teams, 13 have experience developing software, and four others have been business analysts. 23 of the respondents have served in a managerial role (Scrum Masters, Product Managers, Technical Product Managers, Development Managers). Also, 5 others are currently involved in agile research.

Figure 4.7 shows the levels of expertise of the respondents. The practitioners were asked to rate their level of familiarity with the agile software development paradigm on a scale of 1 – 7 with 1 indicating minimal experience and 7 denoting the practitioner to be an expert. A majority of 12 practitioners have stated their level of familiarity with agile software development to be 5, which can be interpreted as a moderate level of expertise. Four respondents have indicated that they consider themselves to be experts in the field. From Figure 4.7, we can see that the majority of the respondents have moderate or higher levels of expertise in agile software development.

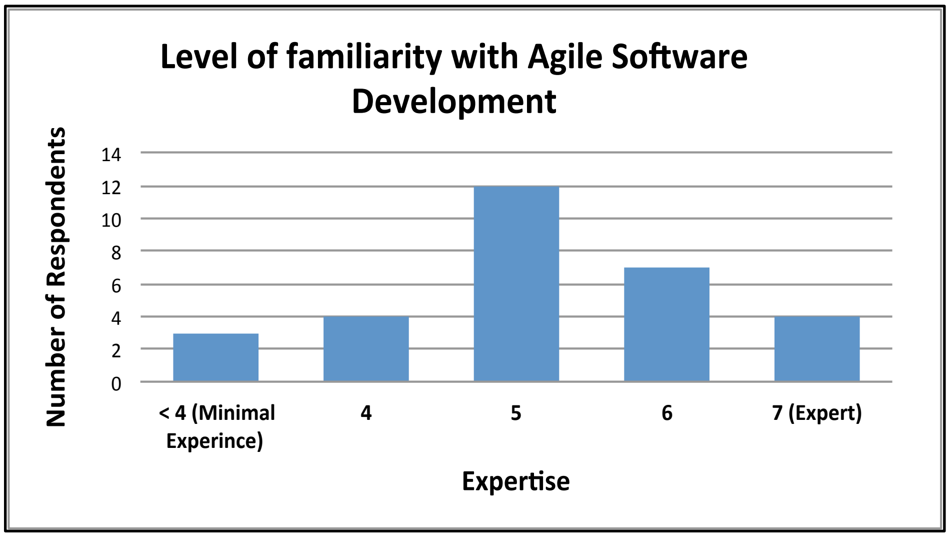


***Figure 4.7. Surveys at the Agile Conference and Company Z - Expertise***

***Need for the assessment approach***

There are many agile assessment approaches available for practitioner use such as the Nokia Test, Scrum Master Checklist, ThoughtWorks Agile Assessment, Comparative Agility to name a few (see Chapter 2). However, these approaches are limited in their scope and application. Our work has been motivated by the need for a comprehensive agile assessment approach that focuses on the people, process, project, and product aspects. To lend credence to our rationale, we asked the following question to the practitioners at the Agile Conference:

> To what extent do you agree or disagree (strongly disagree, slightly disagree, neither disagree nor agree, slightly agree, strongly agree) that there is a *need for a systematic process for assessing the 'goodness' of an Agile Method*.

Figure 4.8 shows the summary of responses to the above question. Out of the 23 respondents, 14 practitioners (60.8% of the respondents) either slightly or strongly agree that there is a need for a systematic approach to assessing agile methods. Six of them neither agree nor disagree and three others disagree that there is a need for an assessment process (see Figure 4.8).

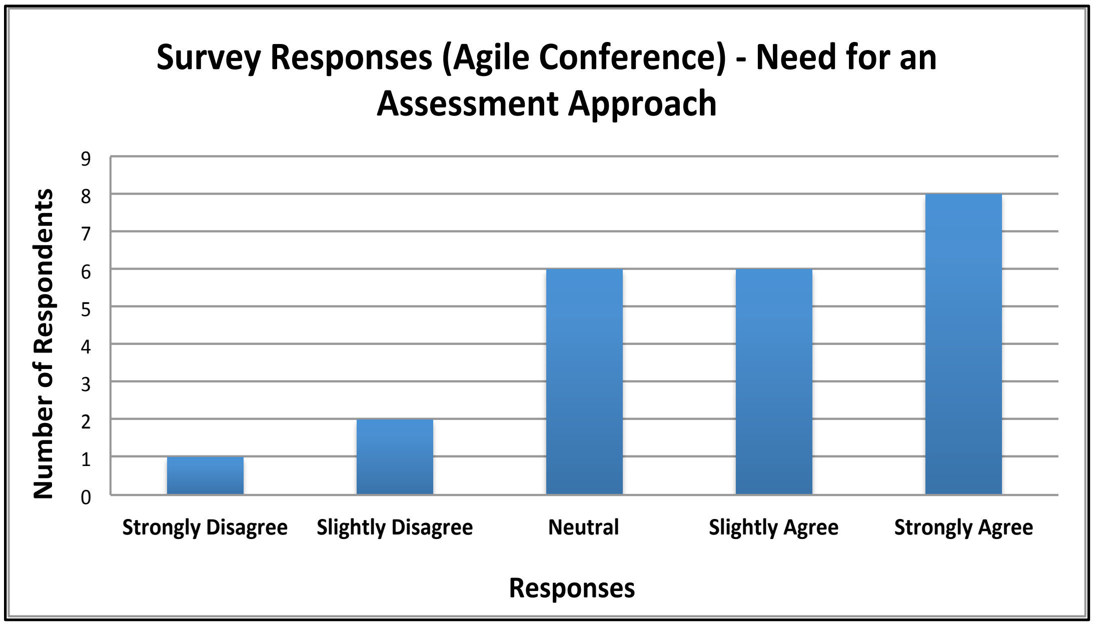


***Figure 4.8. Survey Responses (Agile Conference) - Need for an Assessment Approach***

From the figure above, we see that the majority of the respondents state that there is a need for a systematic approach to assessing agile methods.

As mentioned previously, the survey administered at Company Z was an expanded version of the Agile Conference survey. We asked the practitioners the following question:

> Is there a *need for a systematic process for assessing the 'goodness' of an Agile Method*?
> If not, explain.

Analyzing the responses provided by the seven practitioners and from our discussions with them following the presentation introducing the OPS Framework and assessment methodology, it is evident that the practitioners recognize the need for a systematic approach to agile assessment. The respondents also state that current agile assessment approaches "fall short" and hence, they see the potential for a comprehensive approach.

From the responses gathered from the two surveys, we notice that agile practitioners envision the need for a systematic approach to agile assessment. This substantiates our rationale for the creation of the OPS Framework and assessment methodology. The Framework and the methodology together provide a structured, systematic, and comprehensive approach to assessing agile methods.

***Does the OPS Framework provide guidance?***

The second question that was addressed in the surveys was

> To what extent do you agree or disagree (strongly disagree, slightly disagree, neither disagree nor agree, slightly agree, strongly agree) that as a whole, the OPS Assessment Framework helps *provide guidance to an organization in defining or refining its agile method*.

The work presented in this dissertation is designed to help organizations examine their agile methods and identify the effective components, possible inadequacies, and potential enhancements. Consequently, organizations can refine their adopted agile methods to improve adequacy, capability, and effectiveness. Also, an organization attempting to adopt an agile method can use the OPS Framework to define an agile method that is aligned with organizational and business objectives. Hence, we asked practitioners the above question to gauge the extent to which they agree or disagree that our assessment approach guides organizations in defining or refining its adopted agile methods. Both surveys included the same question.

Among the combined 30 respondents, as shown in Figure 4.9, 17 practitioners (56.66%) state that they slightly or strongly agree that as a whole, the OPS-based assessment approach helps provide guidance to an organization in defining or refining an agile method. Also from Figure 4.9, we see that 13 practitioners neither agree nor disagree with the statement.

It is to be noted that at the time the surveys were administered, the onsite study at Company Z had not been implemented. Hence, the practitioners recorded their responses based only on the presentation delivered and not on a practical implementation of the assessment approach. Though a majority of the respondents agree that the OPS Framework helps provide guidance to organizations, a significant number of them have a neutral view. We conjecture that this is due to the fact that the respondents were unaware of the onsite study at Company Z, which is a practical implementation of our work.

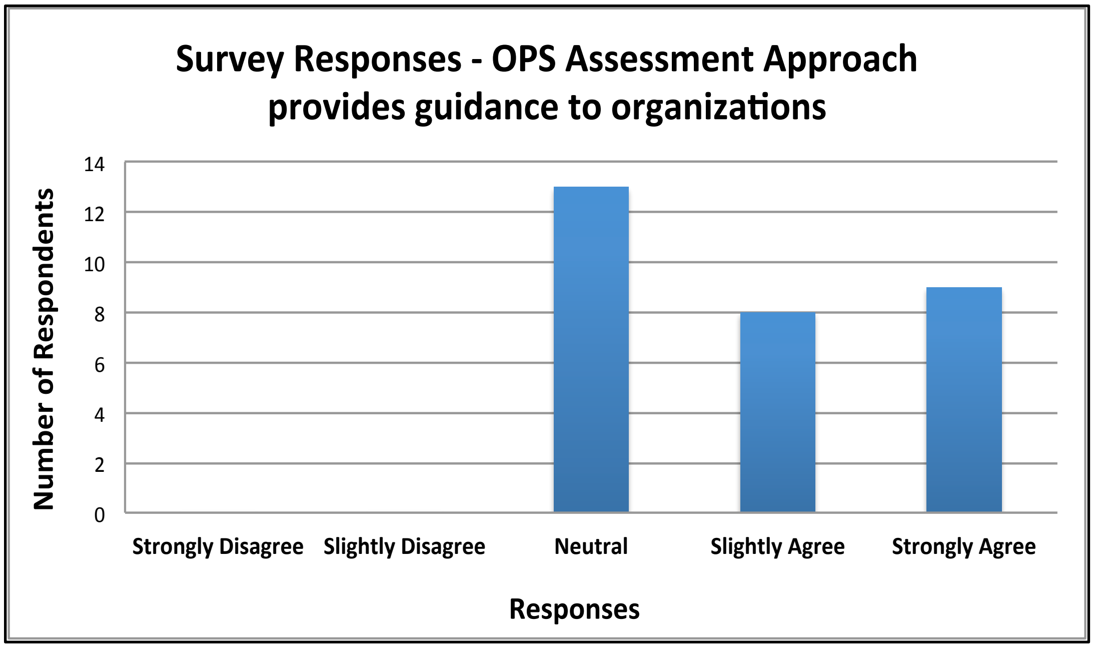


***Figure 4.9. Survey Responses - OPS Assessment Approach Provides Guidance to Organizations***

***Viability of the OPS-based assessment approach***

As discussed previously, by design, our work has been designed to be applied in industry settings. Hence, in addition to gauging practitioner response with respect to the relevance of our work, we recognized that their feedback about the viability of our approach is also important. The following question was asked of the respondents at the Agile Conference:

> To what extent do you agree or disagree (strongly disagree, slightly disagree, neither disagree nor agree, slightly agree, strongly agree) that the OPS Assessment Framework is *a viable approach to assessing an agile method*.

Figure 4.10 summarizes the responses to the above-mentioned question about the viability of the assessment approach. From the figure, we see that 12 practitioners slightly or strongly agree that the OPS approach to assessing agile methods is viable. 11 other respondents neither agree nor disagree with the statement. As discussed previously, these practitioners were not aware of the onsite study at Company Z as the survey was administered before the study was implemented. Some of the practitioners who have completed the survey have noted that they would "like to see a practical application of the assessment approach." An application of the OPS Framework and assessment methodology in an industry setting would substantiate the viability of the assessment approach to a greater extent. Chapter 7 describes our onsite study at Company Z.

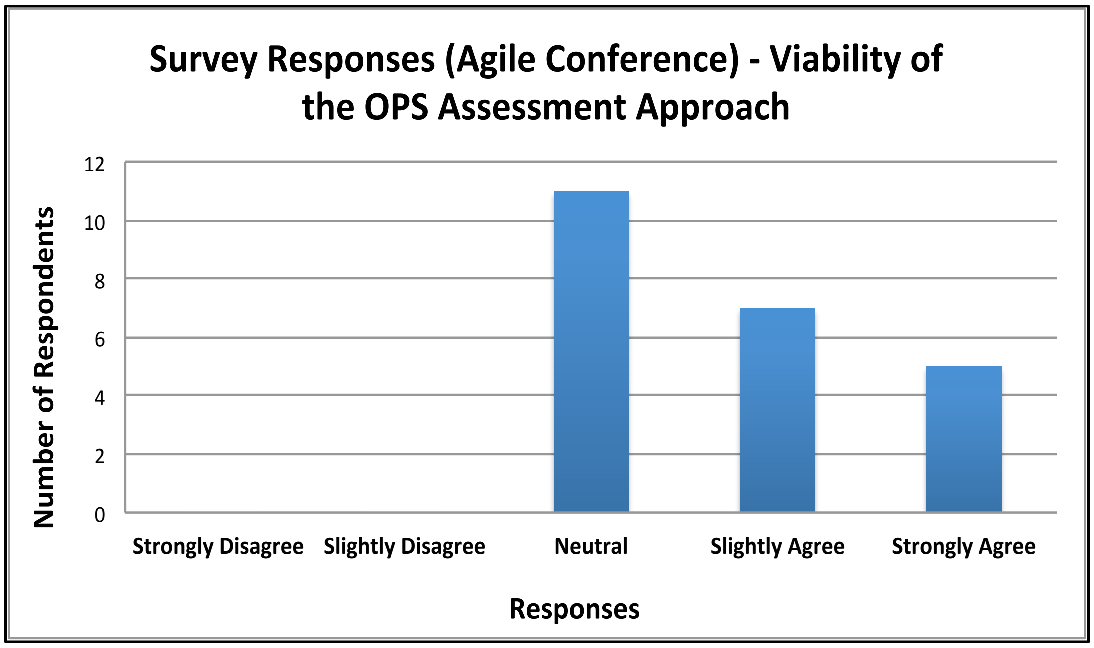


*Figure 4.10. Survey Responses (Agile Conference) - Viability of the Assessment Approach*

In the survey administered at Company Z, the question regarding the viability of the assessment approach was stated as the following:

> To what extent do you agree or disagree (strongly disagree, slightly disagree, neither disagree nor agree, slightly agree, strongly agree) that the OPS Assessment Framework supports a *viable approach to assessing:*
>
> (a) the adequacy of an agile method
>
> (b) the capability of an organization to implement its selected agile method, and
>
> (c) the effectiveness of an agile method

The responses provided by the seven practitioners at Company Z are shown in Figure 4.11. As shown in that figure, a majority of the practitioners state that they slightly or strongly agree that our approach is viable with respect to assessing adequacy. With respect to assessing capability and effectiveness, the personnel at Company Z neither agree nor disagree that our approach is viable. Recall that this survey was administered as a first step to the onsite study at Company Z. The OPS-based assessment approach had not been implemented at that time. Hence, we see that the practitioners did not completely agree that the assessment approach was viable.

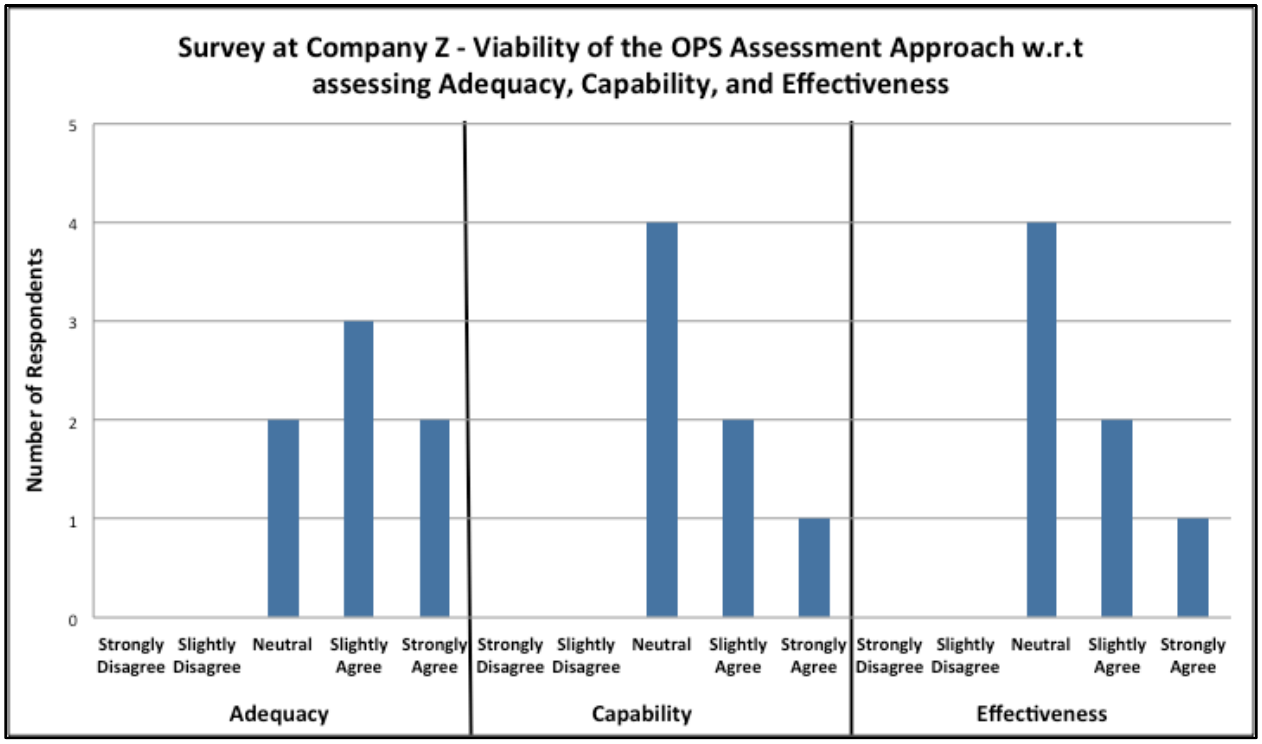


***Figure 4.11. Survey Responses (Company Z) - Viability of the Assessment Approach***

After analyzing the data from the onsite study, we presented them to Company Z personnel (see Chapter 7). During those follow-up meetings, the practitioners confirmed our findings that were obtained by applying the OPS Framework and assessment methodology. From those confirmations, we can infer that Company Z personnel agree that the assessment approach presented in this dissertation is viable.

***Summary***

Our goal in designing and administering the two surveys – one at the Agile 2012 Conference and the other at Company Z, has been to seek confirmation from the agile community regarding the following statements:

(a) there is a need for a systematic approach to assessing agile methods
(b) the OPS Framework and assessment methodology guide an organization in defining or refining its agile method(s), and
(c) the assessment approach is viable.

Analyzing the survey results, we see that practitioners agree that there is a need for a systematic approach to assessing agile methods. With respect to the latter two statements, the practitioners' views have been neutral or confirmatory. This is primarily due to the fact that the presentations did not include details of any practical implementation of the assessment approach. We have since then implemented the OPS

Framework and assessment approach in an industry setting. After the study, Company Z personnel were more strongly convinced that the OPS Framework and assessment methodology are relevant and viable. As mentioned previously, the survey instruments are presented in Appendix D.

## REFERENCES

[1] S. Soundararajan (2011), "A Methodology for assessing Agile Software Development Approaches," *Computing Research Repository,* vol. abs/1108.0427.

[2] S. Soundararajan and J. D. Arthur (2011), "A Structured Framework for Assessing the "Goodness" of Agile Methods," in *18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, pp. 14-23.

[3] S. Soundararajan, J. D. Arthur, and O. Balci (2012), "A Methodology for Assessing Agile Software Development Methods," in *Agile Conference (AGILE) 2012*, pp. 51 - 54.

# 5 Applying the Adequacy Assessment Process

Our substantiation efforts have been two-pronged. That is, we have substantiated (a) the components of the OPS Framework, and (b) the assessment methodology. The results of the former can be found in Chapter 4. Prior to validating the OPS Framework and assessment methodology in Company Z (see Chapter 6), to better understand the applicability of the adequacy assessment process, we have examined the adequacy of three agile methods: XP, FDD, and Method A. XP is a Basis method that most completely reflects the agile philosophy. FDD is advertized as a Basis agile method that is intended for medium to larger scale systems development; it touts a blend of agile values and conventional software engineering principles. Method A is a modified instance of XP, and is currently being used by an organization involved in agile software development. In the discussion in this chapter, we describe the application of our assessment approach to assess the adequacy of XP, FDD, and Method A. We examine their composition, provide observations, and apply the process outlined in Section 3.2.1.

We begin the assessment approach by analyzing the method under consideration. We identify the objectives touted by the method, the supporting principles, and the adopted strategies, and identify the linkages between them. Based on the linkages in the OPS Framework, we can determine the expected set of objectives, principles, and strategies that the method under consideration should tout in order to achieve its stated objectives. If expected components are missing, then the adequacy of the method with respect to achieving its stated objectives is questionable (see Section 3.2.1).

## 5.1 Examining the Adequacy of XP

eXtreme Programming (XP) is designed for small to medium teams which have to develop software quickly in the face of rapidly changing requirements. XP is based on four values: *communication, simplicity, feedback and courage* [1-3] and "it works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation" [3]. The customers are actively involved in the development process. XP came into existence with software development practices found effective over the past two decades.

***Observations about XP***

- The design of the system is usually kept simple.
- Developers work in pairs and implement features using Test Driven Development (TDD).
- The customers write acceptance criteria against which the system is tested at the end of each iteration and release cycle.
- Each iteration lasts 2 - 4 weeks.
- XP is a set of practices that can be adopted by agile practitioners who may choose to use some or all of the practices or may customize the method to suit their needs.
- XP is designed for small teams. However, certain XP practices can be used in larger scale systems development as well (for example, pair programming).

***Objectives***

From our observations, we have identified the set of objectives advocated by XP. Figure 5.1 shows the objectives asserted by XP and the expected set of objectives identified by the OPS Framework. We observe that the two sets are equal, and hence, the objectives of XP are consistent with those of the OPS Framework.

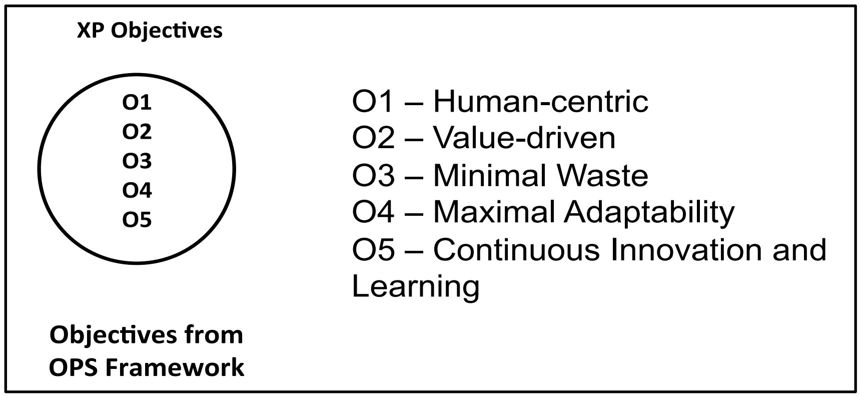

XP Objectives
O1
O2
O3
O4
O5
O1 – Human-centric
O2 – Value-driven
O3 – Minimal Waste
O4 – Maximal Adaptability
O5 – Continuous Innovation and Learning
Objectives from OPS Framework

***Figure 5.1. XP – Expected and Observed Objectives***

***Principles***

Based on the linkages in the OPS Framework, for the objectives touted by XP, the expected set of principles that should be expressed by XP is given in Figure 5.2. In the same figure, the set of principles supported by XP is also shown. More specifically, for the objectives touted by XP, all the nine principles identified by the OPS Framework have to be supported by XP. Comparing the two sets, we observe that the principles asserted by XP are consistent with those identified by the Framework for achieving that method's objectives.

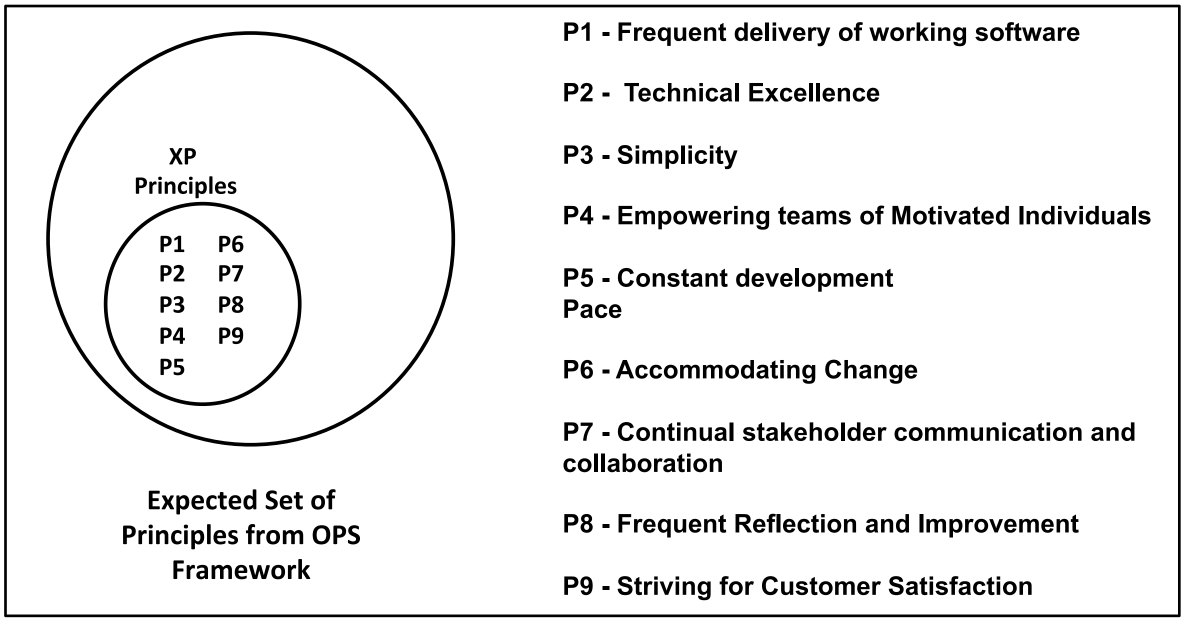

XP Principles
P1 P6
P2 P7
P3 P8
P4 P9
P5
Expected Set of Principles from OPS Framework
P1 - Frequent delivery of working software
P2 - Technical Excellence
P3 - Simplicity
P4 - Empowering teams of Motivated Individuals
P5 - Constant development Pace
P6 - Accommodating Change
P7 - Continual stakeholder communication and collaboration
P8 - Frequent Reflection and Improvement
P9 - Striving for Customer Satisfaction

***Figure 5.2. XP - Expected and Observed Principles***

***Strategies***

From the linkages in the OPS Framework, for the principles expressed by XP, the complete set of strategies provided by the Framework are expected to be asserted by the methodology in order to sufficiently implement its stated principles (see Figure 5.3). XP emphasizes the use of most of the expected strategies. However, the strategies Self-Managing Teams and Configuration Management, which are expected by the OPS Framework for implementing the stated principles, are not advocated by XP. Based on the linkages identified by the Framework, we know that Self-Managing Teams help realize the principles 'Empowering teams of Motivated Individuals' and 'Technical Excellence' (see Appendix B). It is to be noted that the concept of Self-Managing Teams is dependent on the Project Management approach adopted. Hence, when viewing XP independent of organizational objectives, the absence of 'Self-Managing Teams' can be expected.

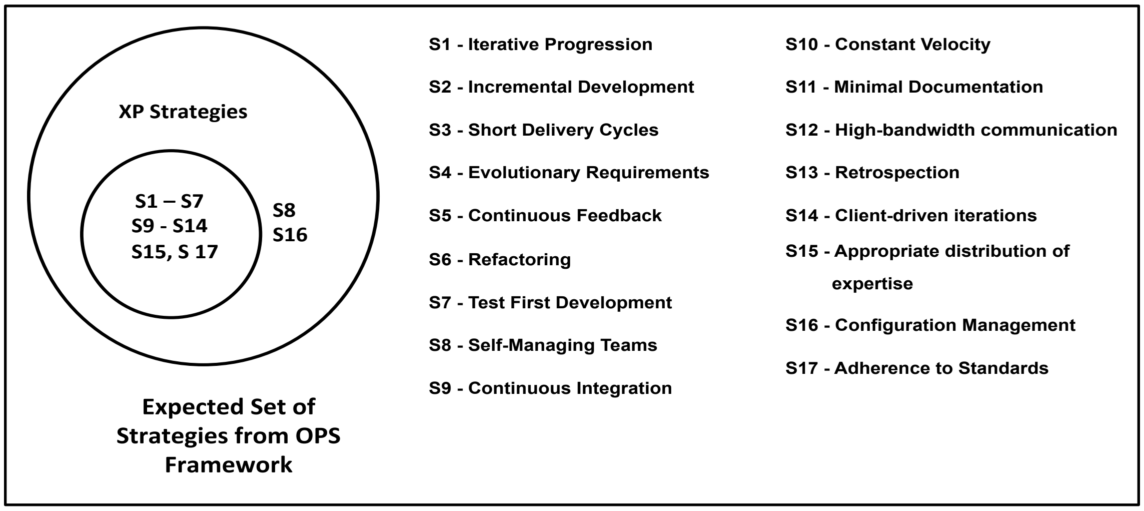

XP Strategies
S1 – S7
S9 - S14
S15, S 17
S8
S16
Expected Set of Strategies from OPS Framework
S1 - Iterative Progression
S2 - Incremental Development
S3 - Short Delivery Cycles
S4 - Evolutionary Requirements
S5 - Continuous Feedback
S6 - Refactoring
S7 - Test First Development
S8 - Self-Managing Teams
S9 - Continuous Integration
S10 - Constant Velocity
S11 - Minimal Documentation
S12 - High-bandwidth communication
S13 - Retrospection
S14 - Client-driven iterations
S15 - Appropriate distribution of expertise
S16 - Configuration Management
S17 - Adherence to Standards

***Figure 5.3. XP - Expected and Observed Strategies***

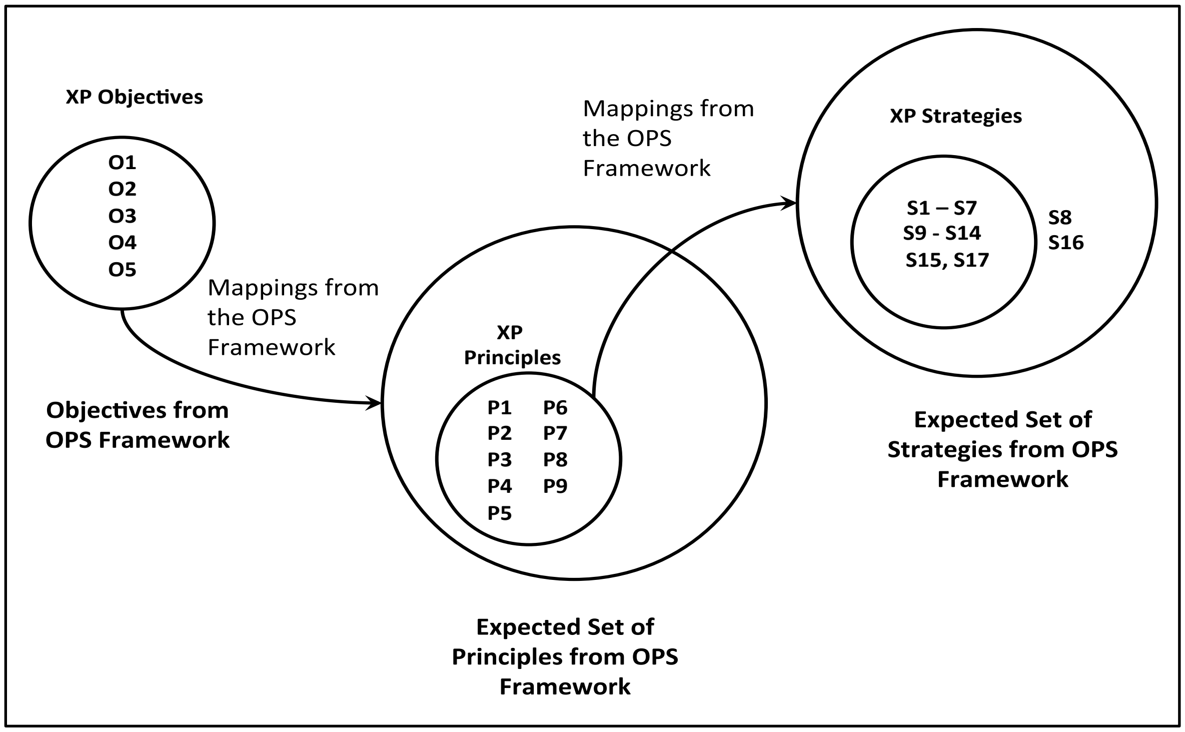

XP Objectives
O1
O2
O3
O4
O5
Objectives from OPS Framework
Mappings from the OPS Framework
XP Principles
P1 P6
P2 P7
P3 P8
P4 P9
P5
Expected Set of Principles from OPS Framework
Mappings from the OPS Framework
XP Strategies
S1 – S7
S9 - S14
S15, S17
S8
S16
Expected Set of Strategies from OPS Framework

***Figure 5.4. XP - Expected and Observed Objectives, Principles, and Strategies***

***Summary***

XP best reflects the agile philosophy. It is designed for small teams. From Figures 5.1, 5.2, and 5.3, we know that XP supports its stated objectives and principles to a large extent. The method is designed to accommodate change even late in the development lifecycle. Figure 5.4 depicts the complete set of objectives, principles, and strategies touted by XP. Based on our assessment, we conjecture that XP is the most adequate with respect to its touted objectives when compared to Method A and FDD.

## *5.2 Examining the Adequacy of FDD*

Feature Driven Development (FDD) is an agile method developed by Jeff De Luca and Peter Coad. It is an iterative and incremental approach that is more structured than the other agile methods and is designed for scalability. Hence, it is commonly used for developing medium to larger scale systems. In the words of Peter Coad, "FDD has just enough process to ensure scalability and repeatability and encourage creativity and innovation all along the way" [2].

FDD does not address the process of requirements engineering. It focuses only on the design and coding phases of development and does not span the complete development process. Therefore, other Requirements Engineering methods have to be used in conjunction with the FDD approach to developing software. Documented requirements in the form of formal requirements specification or use cases are available as input to the FDD process. These requirements identify the scope of the system. The stakeholders then identify features of value to the customers from the documented requirements. The features are defined at a lower level of abstraction. It is mandated that *each feature should take no more than 2 weeks to build*. These features are prioritized and project schedules are defined. A set of features to be developed over the next few days (no longer than 2 weeks) is identified. These features are designed and built iteratively. Software is developed in increments.

***Observations about FDD***

- The success of FDD, like other agile methods, is largely due to recognizing software development as a human endeavor. Hence, 'good' people become the primary asset to the organization adopting FDD. The creators of FDD are of the opinion that the people are more important than the process.
- FDD is diametrically opposite to the other agile methods in its practice of upfront modeling. The other methods, XP for example, declare that the architecture of a system should emerge from an iterative process over several increments. FDD on the other hand places emphasis on "getting it right

the first time"[2, 4]. Hence, during the initial modeling phase, a lightweight architecture is created and the class skeletons are written.

- The major disadvantage of the upfront modeling and design efforts is that these practices reduce the ability to accommodate change later in the development cycle. The architecture of the system is 'frozen' at the beginning of the development effort, and accommodating major changes would be expensive. However, *the fact that FDD supports upfront modeling and design is the primary reason for its suitability for developing medium and larger scale systems*.
- The FDD approach to developing software is not "lean" [5]. More specifically, it proposes to build more than what is needed.
- FDD does not accommodate client driven iterations. Proponents of FDD believe in the development team driving the iterations as opposed to the other agile methods where the clients stipulate the order in which the features are developed.

Based on our analysis and observations, we have identified a set of objectives, principles, and strategies that we conjecture are touted by FDD.

***Objectives***

Among the objectives touted by FDD, 'Value-driven' and 'Continuous Innovation and Learning' are supported to a greater extent than the others. As mentioned previously, FDD does not focus on 'being lean'. That is, FDD does not state the objective of 'Minimal waste' and its associated principle of 'Simplicity'. FDD is intended for medium to larger scale systems development and is more structured than the other agile methods. More specifically, its construction and design minimizes the ability to accommodate change. Hence, FDD supports 'Maximal Adaptability' to a lesser extent. In Figure 5.5, the set of objectives touted by FDD are shown. Minimal Waste is not enunciated by the method and hence, does not feature among the set of stated objectives.

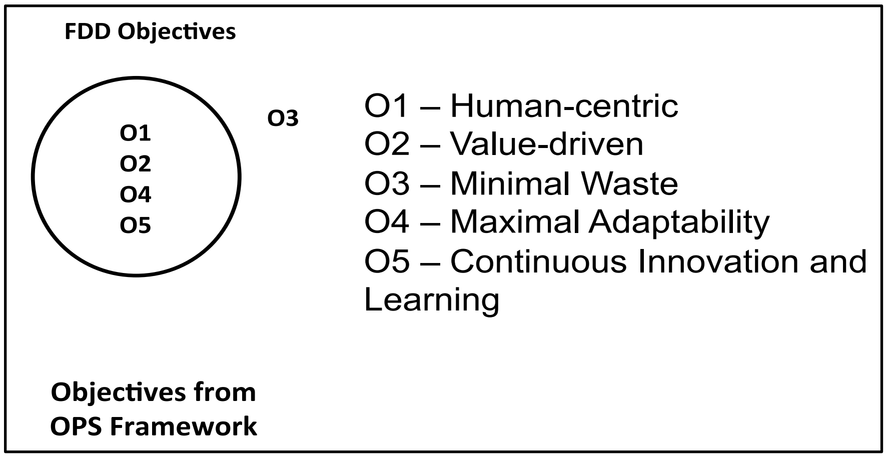

FDD Objectives
O1
O2
O4
O5
O3
O1 – Human-centric
O2 – Value-driven
O3 – Minimal Waste
O4 – Maximal Adaptability
O5 – Continuous Innovation and Learning
Objectives from OPS Framework

*Figure 5.5. FDD - Expected and Observed Objectives*

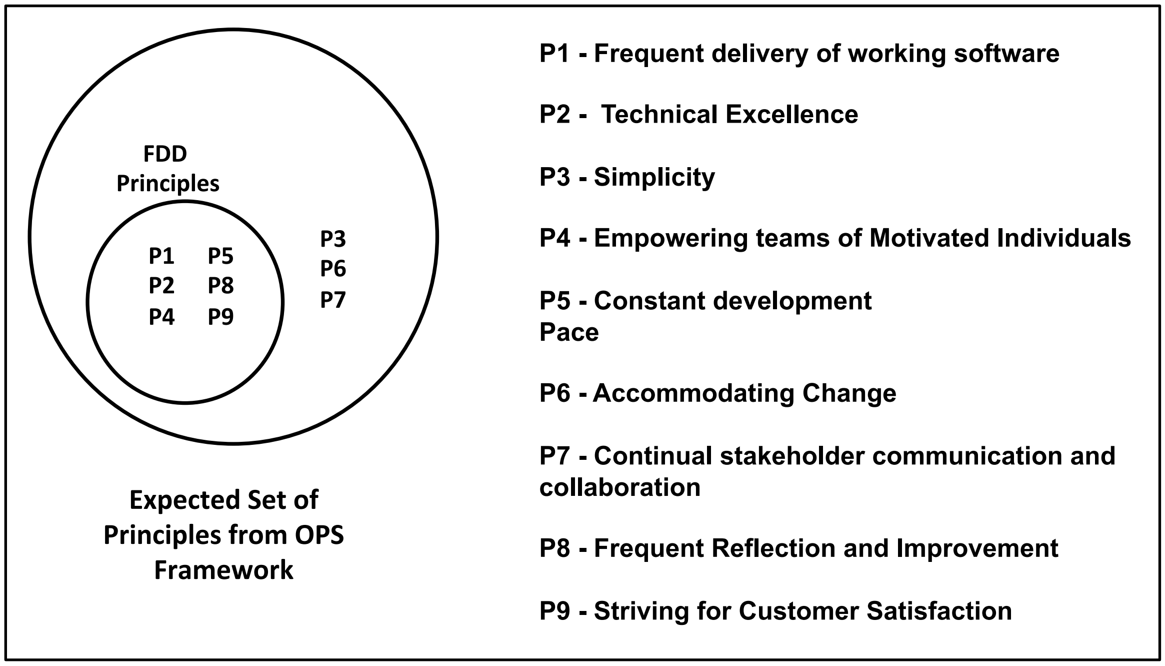

FDD Principles
P1 P5
P2 P8
P4 P9
P3
P6
P7
Expected Set of Principles from OPS Framework
P1 - Frequent delivery of working software
P2 - Technical Excellence
P3 - Simplicity
P4 - Empowering teams of Motivated Individuals
P5 - Constant development Pace
P6 - Accommodating Change
P7 - Continual stakeholder communication and collaboration
P8 - Frequent Reflection and Improvement
P9 - Striving for Customer Satisfaction

*Figure 5.6. FDD - Expected and Observed Principles*

### *Principles*

FDD is an iterative and incremental software development approach that focuses primarily on 'Frequent Delivery of Working Software' and touts necessary strategies that are reflective of the said principle. As mentioned previously, customers are not involved throughout the development lifecycle and the iterations are not client-driven. That is, 'Striving for Customer Satisfaction' is supported to a lesser extent. Teams maintain a constant development pace. At regular intervals, the stakeholders meet to reflect and ‘fine-tune’ their process. FDD, by design, accommodates change to a lesser extent. Figure 5.6 shows the expected and observed principles for FDD. From the linkages in the OPS Framework (see Appendix B), for the objectives stated by FDD, the method is expected to support all the nine principles (see Figure 5.6). However, as discussed above, FDD does not enunciate ‘Simplicity’, ‘Accommodating Change’ and ‘Continuous Stakeholder Communication and Collaboration.’ Hence, the support for achieving the stated objectives is minimized.

### *Strategies*

From the linkages in the OPS Framework, the expected set of strategies for FDD is shown in Figure 5.7. FDD is not expected to implement ‘Minimal Documentation.’ The remaining 16 strategies are expected. However, FDD does not implement the strategies ‘Client-driven Iterations’, ‘Refactoring’, ‘Client-driven Iterations’, and ‘Test First Development.’ Additionally, the requirements are identified upfront, and hence, ‘Evolutionary Requirements’ is not supported. As discussed previously, a continuous feedback loop is maintained to a lesser extent. From the observed set of strategies, we infer that the enunciated principles are achieved to a lesser extent.

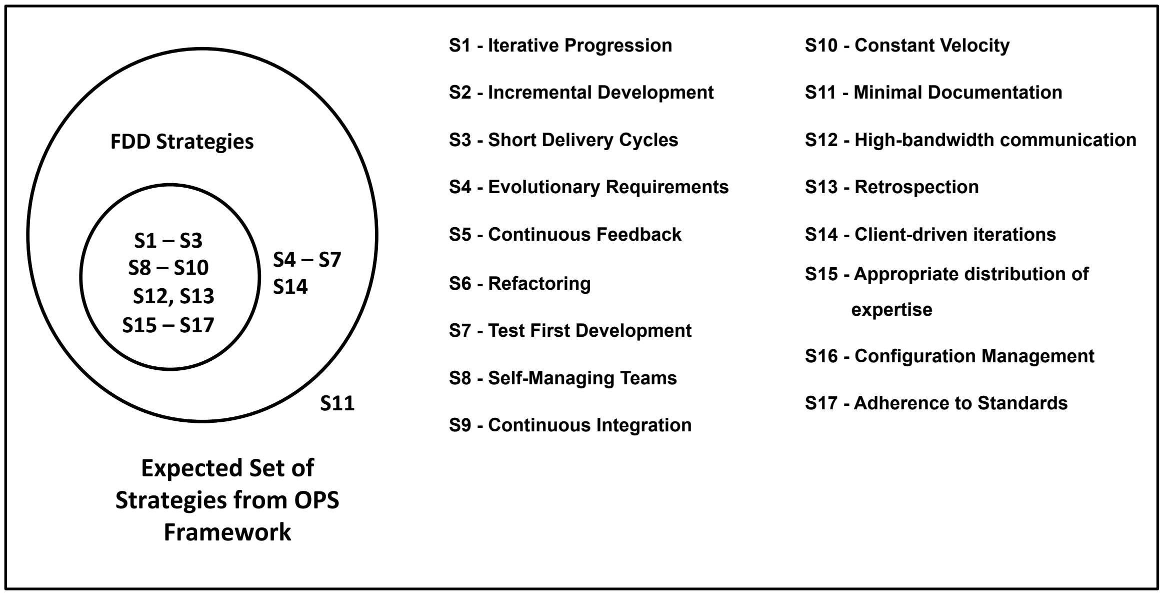

FDD Strategies
S1 – S3
S8 – S10
S12, S13
S15 – S17
S4 – S7
S14
S11
Expected Set of Strategies from OPS Framework
S1 - Iterative Progression
S2 - Incremental Development
S3 - Short Delivery Cycles
S4 - Evolutionary Requirements
S5 - Continuous Feedback
S6 - Refactoring
S7 - Test First Development
S8 - Self-Managing Teams
S9 - Continuous Integration
S10 - Constant Velocity
S11 - Minimal Documentation
S12 - High-bandwidth communication
S13 - Retrospection
S14 - Client-driven iterations
S15 - Appropriate distribution of expertise
S16 - Configuration Management
S17 - Adherence to Standards

*Figure 5.7. FDD - Expected and Observed Objectives, Principles, and Strategies*

***Summary***

As discussed previously, FDD is more suited for medium to larger scale systems. It is more structured than the other agile methods. By design, the method does not tout 'Minimal Waste' and it strives to achieve 'Maximal Adaptability' to a lesser extent. Also, certain classic agile strategies such as 'Client-driven Iterations', 'Refactoring', 'Client-driven Iterations', and 'Test First Development' are not supported. Hence, some of the stated objectives and principles are achieved to a lesser extent. Figure 5.8 shows the set of expected and observed objectives, principles, and strategies for FDD. From our adequacy assessment results for FDD, XP and Method A, we surmise that FDD is the least adequate with respect to its stated objectives than the other two methods.

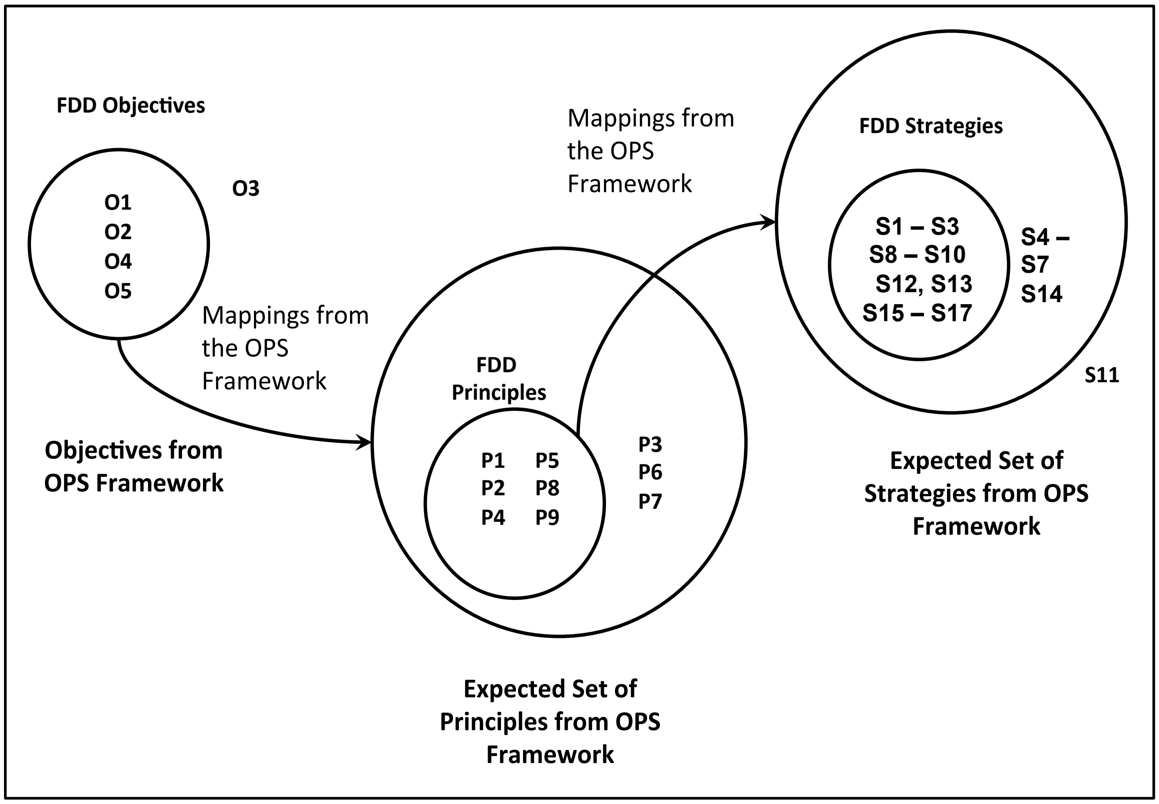

FDD Objectives
O1
O2
O4
O5
O3
Mappings from the OPS Framework
Objectives from OPS Framework
Mappings from the OPS Framework
FDD Principles
P1 P5
P2 P8
P4 P9
P3
P6
P7
Expected Set of Principles from OPS Framework
FDD Strategies
S1 – S3
S8 – S10
S12, S13
S15 – S17
S4 – S7
S14
S11
Expected Set of Strategies from OPS Framework

*Figure 5.8. FDD - Expected and Observed Objectives, Principles, and Strategies*

## 5.3 Examining the Adequacy of Method A

Organization X is a Fortune 1000 company that provides software development, agile coaching and IT consulting services. The method being considered is designated Method A, and is a proprietary agile method of Organization X. As mentioned previously, Method A is a variant of XP. It embodies most of the values and principles stated by XP. However, the culture of the organization precludes the adoption of certain principles and practices. A former employee of Organization X who had used Method A for over two years provided us with the information outlined below.

### *Observations about Method A*

With respect to the objectives, Method A primarily focuses on being human-centric and delivering value to all the stakeholders involved. Method A includes continuous learning and innovation as one of its objectives but, in reality, the teams do not support this objective.

- Significant emphasis is placed on delivering working software at regular intervals, maintaining a constant development pace, continual stakeholder communication and collaboration, and striving to satisfy the customers. Simplicity and frequent reflection and learning are principles that have the least support.
- XP mandates that customers be co-located in order to achieve face-to-face communication and collaboration at any given time. However, co-location is not always feasible. Hence, to overcome this nuance, Method A stipulates that customers must be present during the first and last days of an iteration. Subsequent communication with the customer is achieved via means such as video conferencing.
- Method A mandates the adoption of Continuous Integration.
- Adopting Test First Development (TFD) is optional. Hence, the team members may or may not write the tests before writing code.
- The teams are not self-managing.

***Objectives***

From the above observations, we have derived the set of objectives publicized by Method A (Figure 5.9). As mentioned previously, the teams do not support the objective 'Continuous Innovation and Learning.' Hence, in Figure 5.9, we see that Method A touts only four of the five objectives identified by the OPS Framework.

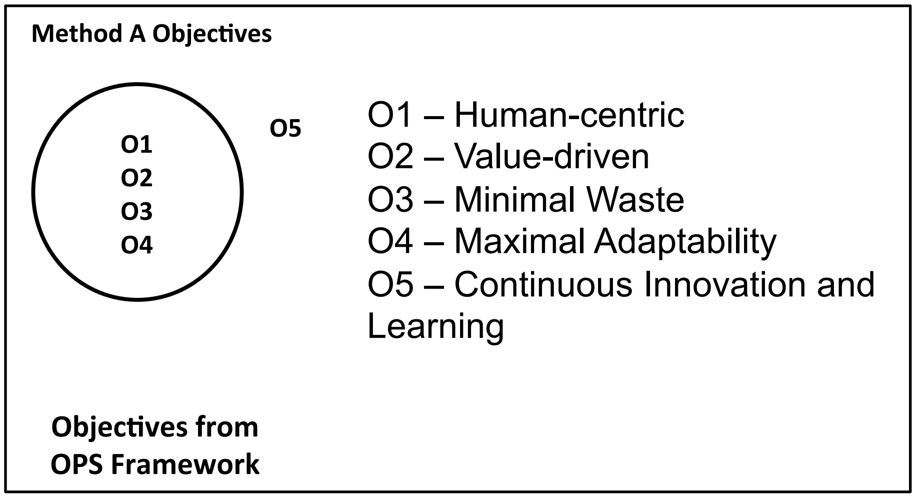

Method A Objectives
O1
O2
O3
O4
O5
O1 – Human-centric
O2 – Value-driven
O3 – Minimal Waste
O4 – Maximal Adaptability
O5 – Continuous Innovation and Learning
Objectives from OPS Framework

*Figure 5.9. Method A - Expected and Observed Objectives*

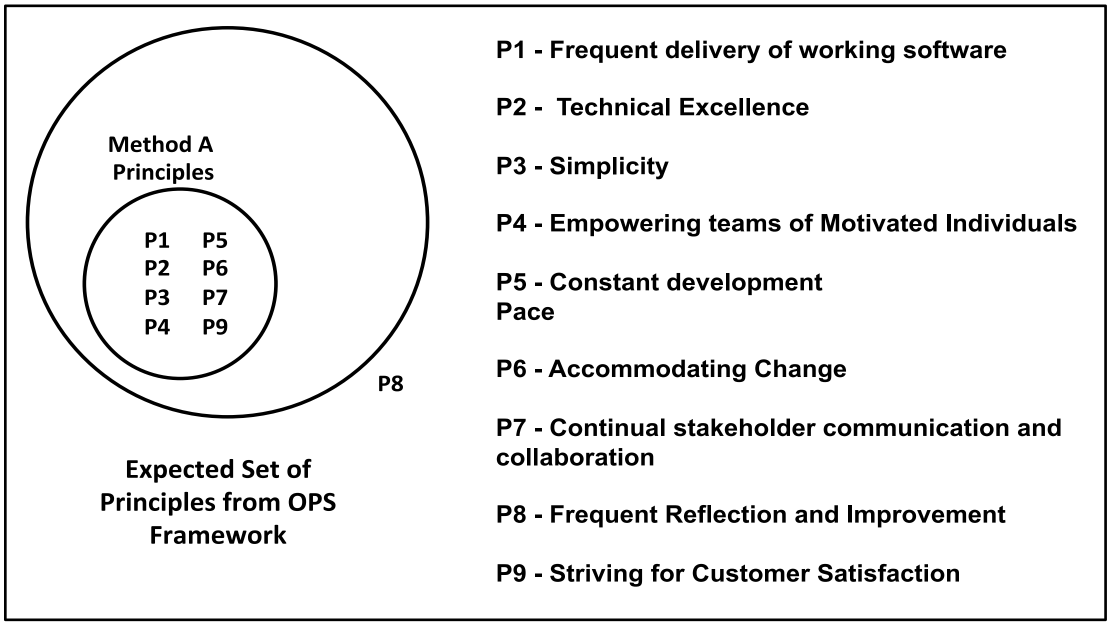

Method A Principles
P1 P5
P2 P6
P3 P7
P4 P9
P8
Expected Set of Principles from OPS Framework
P1 - Frequent delivery of working software
P2 - Technical Excellence
P3 - Simplicity
P4 - Empowering teams of Motivated Individuals
P5 - Constant development Pace
P6 - Accommodating Change
P7 - Continual stakeholder communication and collaboration
P8 - Frequent Reflection and Improvement
P9 - Striving for Customer Satisfaction

*Figure 5.10. Method A - Expected and Observed Principles*

***Principles***

For the objectives that are asserted by Method A, and from the linkages in the OPS Framework, we expect that all the nine principles identified by the Framework be expressed by Method A. Figure 5.9 shows the expected set of principles. From our observations, however, it is evident that Method A does not express 'Frequent Reflection and Learning' as one of its principles (Figure 5.9). Hence the principles supported by Method A, are incomplete with respect to the expected set produced from the linkages in the Framework.

***Strategies***

Relative to strategies, Method A does not advocate Test First Development (TFD), Self-Managing Teams nor Adherence to Standards. Other strategies such as Constant Velocity, Retrospection, Short Delivery Cycles, and Appropriate Distribution of Expertise, are touted minimally. Based on linkages defined by the OPS Framework, we would expect the presence of a set of strategies that includes the above mentioned strategies. In Figure 5.11, we see that these strategies are not implemented by Method A. Hence, given the absence of three strategies and the minimal usage of four others as discussed above, *we question the extent to which Method A sufficiently supports the implementation of its stated principles*.

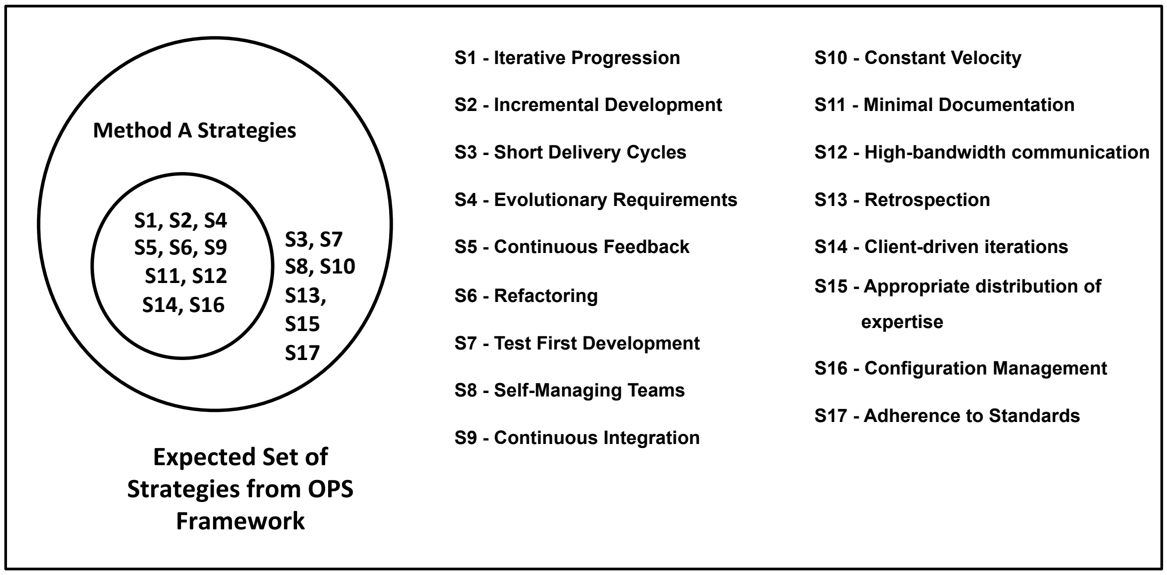

Method A Strategies
S1, S2, S4
S5, S6, S9
S11, S12
S14, S16
S3, S7
S8, S10
S13,
S15
S17
Expected Set of Strategies from OPS Framework
S1 - Iterative Progression
S2 - Incremental Development
S3 - Short Delivery Cycles
S4 - Evolutionary Requirements
S5 - Continuous Feedback
S6 - Refactoring
S7 - Test First Development
S8 - Self-Managing Teams
S9 - Continuous Integration
S10 - Constant Velocity
S11 - Minimal Documentation
S12 - High-bandwidth communication
S13 - Retrospection
S14 - Client-driven iterations
S15 - Appropriate distribution of expertise
S16 - Configuration Management
S17 - Adherence to Standards

***Figure 5.11. Method A - Expected and Observed Strategies***

***Summary***

Method A is an instance of XP. Its application within an organization precludes the adoption and use of certain principles and practices. Comparing the adequacy assessment results of XP (Figures 5.1, 5.2, and 5.3) and Method A (Figures 5.9, 5.10, and 5.11), we can observe the differences in their stated objectives and principles. Method A does not support the objective 'Continuous Innovation and Learning' and its associated principle 'Frequent Reflection and Improvement'. Also, the culture and values of Organization X precludes the implementation of certain strategies as shown in Figure 5.11. The complete set of expected and observed objectives, principles, and strategies are shown in Figure 5.12. Our assessment of the adequacy of Method A is consistent with the perceptions of the agile practitioner who worked at Organization X.

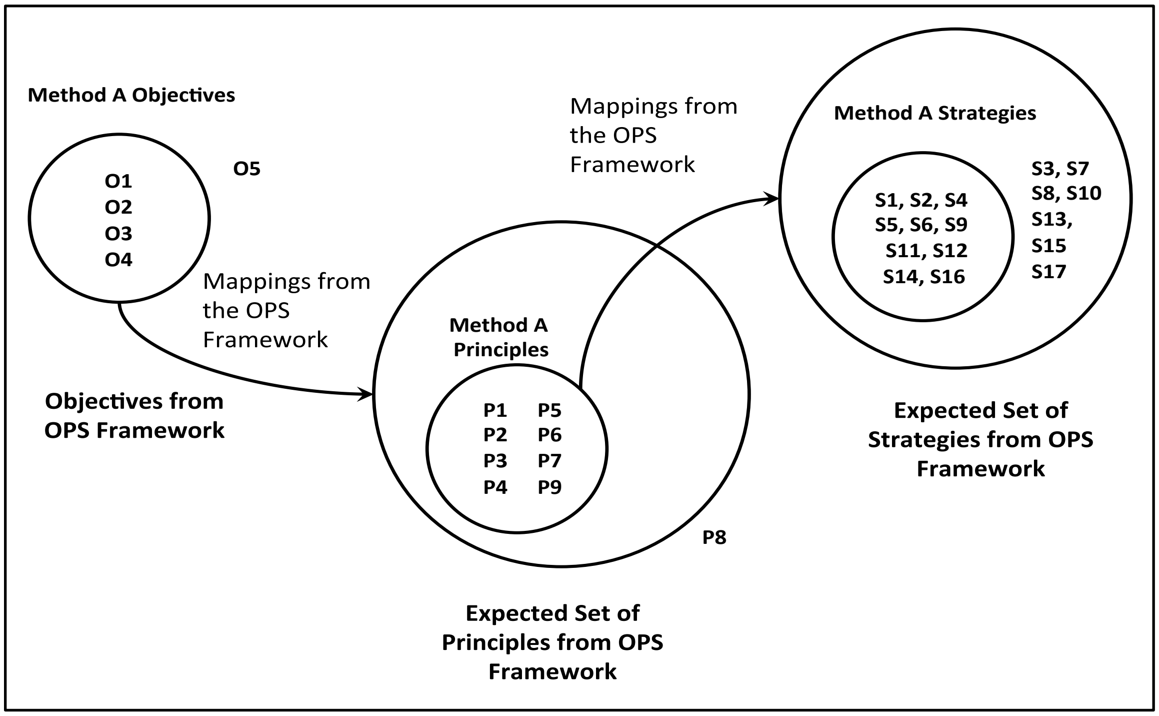

Method A Objectives
O1
O2
O3
O4
O5
Mappings from the OPS Framework
Objectives from OPS Framework
Method A Principles
P1 P5
P2 P6
P3 P7
P4 P9
P8
Expected Set of Principles from OPS Framework
Mappings from the OPS Framework
Method A Strategies
S1, S2, S4
S5, S6, S9
S11, S12
S14, S16
S3, S7
S8, S10
S13,
S15
S17
Expected Set of Strategies from OPS Framework

*Figure 5.12. Method A - Expected and Observed Objectives, Principles, and Strategies*

## *5.4 Summary of Adequacy Assessment Results*

We have applied the top-down, adequacy assessment process to XP, FDD, and Method A. Using the OPS Framework and guided by the three questions given in Section 4.1, we arrive at the conclusions stated below.

- XP best reflects the agile philosophy. It is designed for small teams. XP supports its stated objectives and principles to a large extent. The method is designed to accommodate change even late in the development lifecycle. Based on our assessment, we conclude that XP is the most adequate with respect to achieving its touted objectives when compared to Method A and FDD.
- Method A is an instance of XP. Its application within an organization precludes the adoption and use of certain principles and practices. Comparing the adequacy assessment results of XP and Method A, we can observe the differences in their stated objectives and principles. Method A does not support the objective 'Continuous Innovation and Learning' and its associated principle 'Frequent Reflection and Improvement'. The other touted objectives are maximally achieved. Method A does not advocate some strategies such as TFD, Self-Managing Teams, Adherence to Standards, etc. Consequently, some of the principles are supported to a lesser extent.
- As discussed previously, FDD is more suited for medium to larger scale systems. It is more structured than the other agile methods. By design, FDD supports certain objectives such as 'Minimal Waste' and 'Maximal Adaptability' minimally. Also, strategies such as 'Client-driven Iterations', 'Refactoring', and 'Test First Development' are not supported. Hence, some of the touted objectives and principles are achieved to a lesser extent.

In a relative sense, we have concluded the following:

1. XP is more adequate than FDD or Method A with respect to supporting its objectives, and
2. Method A is more adequate than FDD with respect to supporting its objectives.

These results are consistent with our observations, but further independent confirmation by Subject Matter Experts (SMEs) is still needed.

## REFERENCES

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta (2002), *Agile Software Development Methods: Review and Analysis*. Finland: VTT Publications.

[2] J. Highsmith (2002), *Agile Software Development Ecosystems*: Addison-Wesley.

[3] R. Jeffries (2002), *What is Extreme Programming*, http://xprogramming.com/what-is-extreme-programming

[4] S. R. Palmer and J. M. Felsing (2002), *A Practical Guide to Feature Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR.

[5] A. Qumer and B. Henderson-Sellers (2008), "An evaluation of the degree of agility in six agile methods and its applicability for method engineering," *Information and Software Technology,* vol. 50, pp. 280-295.

# 6 Study at Company Z

We recognize that in order to effectively substantiate our assessment process, and determine the viability of our approach, the OPS Framework and assessment methodology must be applied within an organization that adopts agile methods. We worked with Company Z, which is located in Blacksburg, Virginia, to substantiate our work. Company Z delivers web-based applications to businesses around the world. Started in the late 1990's, it has since grown to serve an extensive customer base. The company employs about 4500 (full-time + part-time) personnel.

## *6.1 Study Objectives and Benefits*

This study has been designed to be mutually beneficial to Company Z and to the research presented herein. It provides Company Z with insights concerning its approach to Agile Software Development, and the author with a validation site. Our objective is to examine several of Company Z's agile methods and identify the effective components, possible inadequacies, and potential enhancements. Our goals in designing and conducting this study at Company Z are to

(a) Elicit additional feedback concerning the components and structure of the OPS Framework,

(b) Perform a *capability assessment* on one or more agile teams, and

(c) Take a cursory look at the effectiveness of the adopted agile methods.

The latter two necessarily imply an adequacy assessment on the respective agile methods. Following the adequacy and capability assessment, we then present our findings to members of that organization. The intent is to determine (a) to what extent those findings are consistent with perceptions, and ultimately, (b) the validity of our results.

## *6.2 Study Mechanics*

The study is a three-step process as outlined below:

Step 1: *Introduce Assessment Framework and approach*

- A presentation to appropriate Company Z personnel explaining the OPS approach to assessing agile methods, and then soliciting their feedback on the approach

Step 2: *On-site study*

- An on-site examination of two or more agile teams to assess the adequacy of the agile methods being used, the capability of the teams' environments to support their adopted agile methods, and a cursory examination of the effectiveness of those methods. The duration of the on-site examination was three months.
- We examined the agile methods adopted by four agile teams at Company Z. In this dissertation, they are identified as Teams A, B, C, and D.

Step 3: *Feedback*

Present our findings to Company Z and confirm the validity of our results.

All three above-mentioned steps have been completed. In the following sections, we describe the aspects of the study, analysis and results, and details of the follow-up meeting(s) with the teams to validate the results.

## *6.3 Aspects of the Study*

The duration of the on-site examination was three months (September 10, 2012 – December 12, 2012). Our assessments are based on our observations that were recorded during the three-month period. All of the teams in Company Z use agile methods. Given the iterative nature of agile processes, during the above-mentioned timeframe, we can observe the projects at different stages of completion. The study has also been designed to be minimally intrusive.

During the course of our on-site study, we attended daily stand-ups, release and sprint planning meetings, and retrospectives, and observed the development processes. We also held discussions with members of the four teams on topics including but not limited to the following:

- Desired objectives
- Current state of development process
- Issues faced
- Steps taken to solve or mitigate the issues
- Communication and collaboration with the customers
- Indicators of customer satisfaction

Surveys to gather information about the OPS Framework, the Assessment Methodology, and perceptions of the current processes were also administered.

The data collected during the onsite study can be categorized as the following:

- OPS-based observations

  These findings have been guided by the hierarchical structure and components of the OPS Framework, and in particular, the indicators.

- Additional team-specific observations

  These team-specific observations are based on the authors' perceptions of the processes and development environments. It is to be noted that these observations are not a component of the formal OPS assessment process.

- Elements specific to Company Z

  We also observed certain aspects of the process, environment, people, etc. that are specific to Company Z. The implications of these aspects require further discussion and examination.

The OPS-based observations have guided the assessment process. We performed both top-down and bottom-up examinations to assess the adequacy, capability, and effectiveness. Assessing effectiveness is a longitudinal process. Given the short duration of the onsite study, our assessment of effectiveness is preliminary.

## *6.4 The Foundation for Calculating and Interpreting Assessment Scores*

We have examined the capability of Teams A, B, C, and D to support the implementation of their adopted agile methods. We have also ***cursorily*** examined the effectiveness of the agile methods adopted by the above-mentioned four teams. As a first step to assessing capability and effectiveness, we examined the adequacy of the methods adopted by the four teams. Using the adequacy assessment process outlined in Section 3.2.1, we found all of these methods to be ***adequate with respect to achieving their stated objectives***.

To assess the capability and effectiveness, we have constructed a hierarchy of indicators. We have identified the observable properties of the people, process, project, product, and the environment. Based on our observations of the four teams at Company Z, we have also refined the indicators and the questions associated with those indicators. Table 3.5 lists the number of questions used to measure each indicator (i.e. strategy, property pair) from both a capability and effectiveness perspective. The indicator hierarchy includes 80 indicators and 150 questions (see Appendix C).

We first answer each question that has been identified as part of the hierarchy. Each answer is recorded as numeric scores on a pre-determined scale. Then, a weighted average of the answers (numeric scores) to the questions associated with each property is computed. These averages are further propagated to the strategies level and we compute a weighted average of the scores of all the properties associated with each strategy. Following the same process, these weighted averages are propagated from the strategies level to the objectives. For the computations, we have assigned equal weights to all the elements. To summarize, at each level of the hierarchy of indicators, a weighted average of the elements at the level below is computed. Then, these averages are propagated to the next higher level in the hierarchy and a final score is obtained at the root level.

### Measurement Scales for Computing OPS Scores

This section describes the OPS based scoring process used for assessing capability and effectiveness. As described previously, we use the hierarchy of indicators to propagate the scores from the questions level to the objectives. We have constructed two hierarchies of indicators, one each for capability and effectiveness. Appendix C provides the complete hierarchies used in the assessment process.

## *Capability: The Measurement Scale*

We define the capability of an organization as its ability to provide the supporting environment conducive to the implementation of an agile method. In assessing the capability of an organization, we are concerned with the characteristics of its *internal environment*. The internal environment of an organization is primarily composed of its resources and competencies. More specifically, in an organization, the characteristics of its people, the process that it adopts, its environment, and its projects are reflective of the characteristics of its internal environment. Hence, we use observable properties of the environment, people, process and project in our assessment of capability. For example, the presence of open physical environments in an organization is *indicative* of the organization's capability to foster face-to-face stakeholder communication and collaboration at any given time.

To summarize, Capability answers questions concerning the existence of an organization's or team's resources. Hence, the answers to the questions in the hierarchy are usually binary. We recognize that for some of the questions, our answers would lie somewhere in between the extremes. Consequently, we have used a ternary scale to answer the questions associated with the indicators. The scale is shown in Table 6.1.

***Table 6.1. Scale used for answering questions from a Capability perspective***

| **Answer** | **Score** |
|---|---|
| Yes/Fully | 5 |
| Partially | 3 |
| No/Marginally | 1 |

We answer the questions, assign scores based on the scale given in Table 6.1, compute weighted averages, and finally propagate those scores to the higher levels in the hierarchy. At the root level, we get a score on a five-point scale for Capability. For capability, we recommend that the final scores obtained at each hierarchical level be interpreted using the scale given in Table 6.2.

***Table 6.2. Scale for interpreting final scores***

| Score Range | Interpretation |
|---|---|
| 4.5 – 5.0 | Maximal |
| 3.5 – 4.4 | Considerable |
| 2.5 – 3.4 | Moderate |
| 1.5 – 2.4 | Somewhat |
| 1 – 1.4 | Marginal |

For example, if the Capability score for Team X is 3, we can conclude that the team is ***Moderately Capable*** of supporting/ implementing the strategies, principles, and objectives of its adopted agile method.

## *Effectiveness: The Measurement Scale*

An agile method is judged effective if it produces "*working software*" that is a *quality product*, *on time*, *within budget*, and *of value to the customers*. Assessing the effectiveness of the agile method involves the identification of properties of process artifacts and the product, which focus on the system produced and its value to the stakeholders. More specifically, Effectiveness answers questions concerning the extent to which the desired outcomes were achieved. We recognize that there can be degrees to which a team is effective. Hence, to answer questions pertaining to effectiveness, we have used a 5-point likert scale shown in Table 6.3.

***Table 6.3. Scale used for answering questions from an Effectiveness perspective.***

| Answer | Score |
|---|---|
| Maximally | 5 |
| Considerably | 4 |
| Moderately | 3 |
| Somewhat | 2 |
| Marginally | 1 |

Similar to the Capability Assessment process, we compute weighted averages at each level, and propagate those scores to the higher levels in the hierarchy. At the root level, we get a score on a five-point scale for

Effectiveness. We recommend that the scores for effectiveness also be interpreted using the scale given in Table 6.2.

***Assessing the effectiveness of agile methods is a longitudinal process***. Given the short duration of the study, we have completed a preliminary assessment of effectiveness of the methods adopted by Teams A, B, D, and C. We possess limited information about the effectiveness of the strategies. This is especially true for four of the strategies – *Test First Development, Refactoring, Continuous Integration, and Configuration Management*. Though we have included these strategies in the assessment, we have removed some of the relevant questions since we do not possess sufficient information to answer those questions. Our assessment of the effectiveness is preliminary.

## *6.5 Scoring Sources*

In this chapter, we have included three sets of scores. In addition to the OPS based scoring, we have included scores assigned by Subject Matter Experts (SMEs) and Company Z personnel. The authors of this report are the SMEs. These additional scores have been recoded for both capability and effectiveness. This section provides an overview of each of these scoring sources.

### 6.5.1 Scores derived from OPS Computations

The scores derived from the OPS computations are indicator based (see Section 3.5). A set of questions is associated with each indicator to help measure the associated OPS strategy. We then propagate the scores from the strategies level to the objectives. Recall that at each level of the hierarchy, we compute the weighted averages of the scores at the level below it. In our computations, all ***the linkages and associations in the OPS Framework are assigned equal weights***. The OPS based scores are labeled "Computed Scores" in Section 6.6. Refer Appendix C for the indicator hierarchies.

### 6.5.2 Scores determined by Subject Matter Experts

As mentioned previously, the author of this dissertation and her advisor are the SMEs. Examining the objectives, principles, and strategies from the capability and effectiveness perspectives, we (SMEs) assigned scores for the above-mentioned three components. These scores, labeled "Expert Scores", were determined ***independently of the scores computed using the indicator hierarchy and those elicited from the Company Z personnel***.

For each team, the SMEs have determined the extent to which the organization provides resources, expertise, and appropriate knowledge-base to that team to support the (a) implementation of strategies,

(b) adherence to principles, and (c) achievement of objectives. Additionally, the SMEs have also examined the effectiveness of the agile methods adopted by the four teams. The expert scores are assigned using the scale given in Table 6.3 ***before*** analyzing the gathered data and computing the OPS based scores.

We have included these scores when presenting the results in Section 6.6. In most cases, we found that (a) the differences between the computed scores and our expert scores are negligible, and (b) the two sets of scores follow *similar trends*. Any significant differences in the two sets of scores can more often than not be attributed to the factors discussed in Section 6.6.6.

### 6.5.3 Scores elicited from Company Z personnel

As mentioned previously, during the course of our on-site study, we administered surveys about perceptions of the current processes. We also interviewed key personnel from the four teams. From the surveys and interviews, we have gathered information about the following:

- ***Effectiveness - Strategies Level***: Survey completed by the developers and Quality Engineers from all the four teams.

  In this survey, members of the development team analyzed their agile methods and recorded their perceptions of the effectiveness of each strategy. We have used the scale shown in Table 7.3 to record the practitioners' responses.

- ***Effectiveness – Objectives Level***: Survey completed by the Scrum Masters, Technical Product Managers, Product Owners, and Dev Managers from all the four teams.

  In this survey, the participants were asked to provide their views about the extent to which their teams achieve each of the objectives identified by the OPS Framework. The values were recorded on a 1 – 10 scale. We assigned weights to the scores recorded by each respondent. We then converted these weighted averages to a 1- 5 scale.

- ***Capability – Strategies Level***: Interviews and discussions with members of the different teams.

  Based on our discussions, interviews, and observations of the development processes, we have inferred the capability of the teams to support the implementation of each strategy. We have used the scale shown in Table 6.3 to record the scores.

The objectives and the principles are at higher levels of abstraction than the strategies. Consequently, it is difficult for the practitioners to assign scores for those components. Strategies, on the other hand, are more tangible concepts that are better understood by practitioners. Hence, we primarily focused on strategies when eliciting scores from Company Z personnel. They are labeled "Company Z scores – numeric" in the next section.

In the following section, we provide our team-specific observations, evaluations, and results.

## *6.6 Assessment and Observations*

In this section, we provide the OPS based assessment for Teams A, B, C, and D. Additionally, our team-specific observations have also been included. It is to be noted that these observations are based on our perceptions and are not a component of the formal OPS assessment process.

Sub-sections 6.6.1, 6.6.2, 6.6.3, and 6.6.4 outline the team-specific assessment and observations for Teams A, B, C, and D respectively. In sub-section 6.6.5, we list the observations that are applicable to all four teams. These observations are, in some sense, pertinent to Company Z, Blacksburg. In general, we see that the ***three sets of scores (computed, expert, and Company Z) exhibit similar trends***.

For each team, the assessment includes the following components:

- Computed scores based on the OPS Framework
- Expert scores assigned by the SMEs (the author of this dissertation and her advisor)
- Scores elicited from Company Z personnel
- Differences in scores between
    - Computed and expert scores
    - Computed and the average of Company Z and Expert scores
- Insights concerning substantial differences, if any, between the computed, expert, and Company Z scores.

## 6.6.1 OPS Based Assessment and Observations: Team A

Team A follows the Continuous Flow approach. This method allows them to release software frequently. Due to evolving team needs, they have moved away from a Scrum-like process. This team is involved in developing email and web applications. Table 6.4 outlines Team A's profile.

***Table 6.4. Profile - Team A***

***Team Size***

10

***Roles***

Product Owner (1)

Development Manager (1)

Developers (5)

Quality Engineer (1)

Software Engineer in Test (1)

Intern (1)

***Development Process***

Continuous Flow

***Tools used***

PivotalTracker, Git, RedMine, Wiki, Maven, PhPUnit, Apache Continuum

***Velocity***

6 story points per week.

***Iteration length***

Though they do not follow an iterative process, they plan their work for 2 – 3 weeks. At the end of this period, they schedule a software release.

### *6.6.1.1 Team-specific Assessment*

Based on our OPS observations and assessments, the overall Capability and Effectiveness scores for Team A are:

| | |
|---|---|
| **Capability: 4.395** | **Effectiveness: 4.133** |

Using Table 6.2 to interpret these scores, both values fall in the "***Considerable***" range (3.5 – 4.4). The Capability score is closer in value to the "Maximal" range. Hence, Team A is ***considerably*** (a) capable of implementing its adopted agile method and (b) effective in implementing that method.

#### **ASSESSMENT: ADEQUACY**

Recall that our first step in the assessment process is an examination of the adequacy of the agile method under consideration. The OPS Framework identifies five objectives, nine principles, and 17 strategies. Guided by the OPS Framework, we determine the expected set of principles and strategies that are essential to achieve the stated objectives. Prior to computing the OPS-based capability and effectiveness scores given above, we examined the adequacy of the method adopted by Team A.

As mentioned previously, Team A follows Continuous Flow. An examination of Team A's objectives indicate that the team strives to achieve all the five objectives identified by the OPS Framework. Consequently, the method adopted by Team A is expected to support all the nine principles identified by the Framework. Continuous Flow, by design, does not advocate the notion of a 'Constant Development Pace.' Hence, this principle is not supported. Based on the linkages in the OPS Framework, with the exception of Constant Velocity, Team A's agile approach is expected to implement the remaining 16 strategies. Once again, by design, Continuous Flow does not tout 'Iterative Progression' and 'Retrospection'. Hence, there is a difference in the expected and observed set of strategies. Though the method predominantly supports the expected set of principles and strategies required to achieve the stated objectives, by design of that adopted method, certain principles, and objectives maybe achieved to a lesser extent.

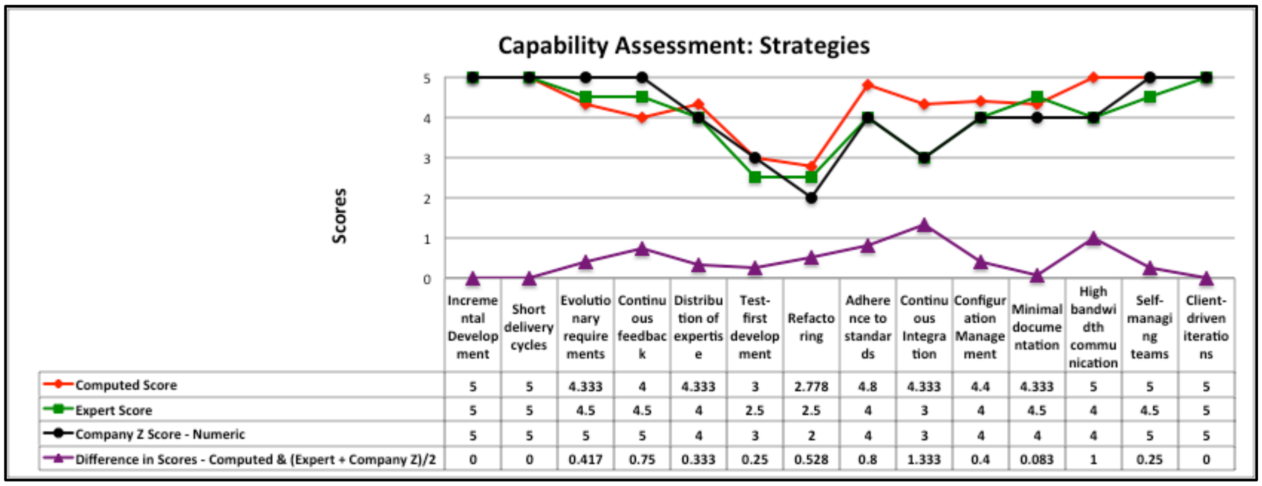
In the assessment of Team A's capability and effectiveness, the principle 'Constant Development Pace', and the strategies 'Constant Velocity', 'Iterative Progression' and 'Retrospection' are not featured. During the follow-up meeting with Team A, the members of this team confirmed our examination of the composition of their adopted agile method. More specifically, the team confirmed our adequacy assessment of their process.

## ASSESSMENT: CAPABILITY

Figures 6.1, 6.2, and 6.3 show the Capability Assessment results for the strategies, principles, and objectives respectively. The numeric values show similar trends for all three components.

### *A Strategies Level Assessment*

Refactoring is implemented to a lesser degree in this team. Hence, the substantially lower scores for that strategy as shown in Figure 6.1. For the strategy Continuous Integration, from Figure 6.1, we see that the scores exhibit similar trends. However, the Expert and Company Z scores are considerably lower. Recall that the linkages in the OPS Framework are equally weighted. On the other hand, the other two sets of scores are implicitly weighted. Hence, we notice the difference in the three sets of scores.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development | Refactoring | Adherence to standards | Continuous Integration | Configuration Management | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ◆ Computed Score | 5 | 5 | 4.333 | 4 | 4.333 | 3 | 2.778 | 4.8 | 4.333 | 4.4 | 4.333 | 5 | 5 | 5 |
| ■ Expert Score | 5 | 5 | 4.5 | 4.5 | 4 | 2.5 | 2.5 | 4 | 3 | 4 | 4.5 | 4 | 4.5 | 5 |
| ● Company Z Score - Numeric | 5 | 5 | 5 | 5 | 4 | 3 | 2 | 4 | 3 | 4 | 4 | 4 | 5 | 5 |
| ▲ Difference in Scores - Computed & (Expert + Company Z)/2 | 0 | 0 | 0.417 | 0.75 | 0.333 | 0.25 | 0.528 | 0.8 | 1.333 | 0.4 | 0.083 | 1 | 0.25 | 0 |

***Figure 6.1. Team A - Capability Assessment: Strategies***

By design, the Continuous Flow approach does not advocate the strategies Iterative Progression and Retrospection. Hence, these two strategies have not been included in the assessment of Team A's method. More specifically, these two strategies were assigned a weight of 0 in order to nullify their impact on the higher level principles.

To validate the computed OPS values at the strategies level, we have determined the Pearson correlation coefficient (see Table 6.5) between the sets of values for (a) the Computed OPS and SMEs, and (b) the Computed OPS and Company Z.

***Table 6.5. Team A – Capability Assessment: Pearson Correlation Coefficient at the Strategies Level***

**Pearson Correlation At the Strategies Level – Capability Assessment**

**Computed OPS and Expert:**

**r** = 0.81061, **N** = 14, **p-value** = 0.00044, two-tails

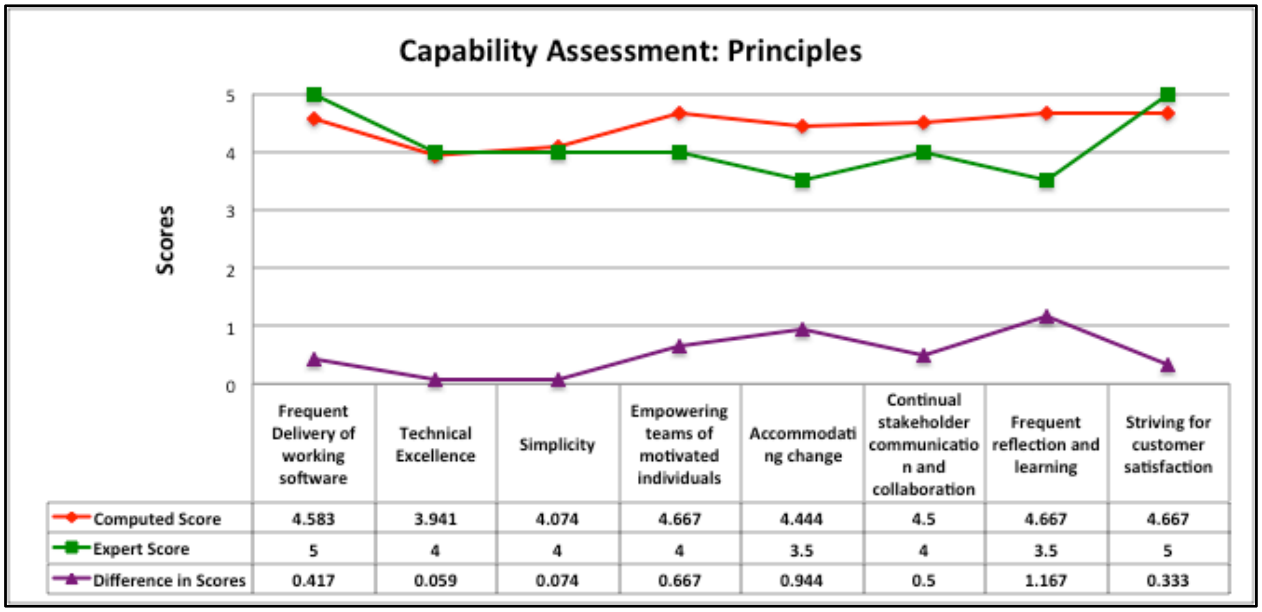
**Computed OPS and Company Z:**

**r** = 0.75335, **N** = 14, **p-value** = 0.00187, two-tails

In both cases given above in Table 6.5, the correlation is positive. Additionally, the results are ***statistically significant***. Hence, we can conclude that the similarity in trends exhibited by the three sets of values is not by random chance. That is, the observed correlation between the Computed and Expert scores, and the Computed and Company Z scores is significant, and not a chance occurrence.

### *A Principles Level Assessment*

In Figure 6.2, we see a significant difference between the computed OPS scores and the Expert scores for the principle Frequent Reflection and Learning.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.583 | 3.941 | 4.074 | 4.667 | 4.444 | 4.5 | 4.667 | 4.667 |
| Expert Score | 5 | 4 | 4 | 4 | 3.5 | 4 | 3.5 | 5 |
| Difference in Scores | 0.417 | 0.059 | 0.074 | 0.667 | 0.944 | 0.5 | 1.167 | 0.333 |

***Figure 6.2. Team A - Capability Assessment: Principles***

By design, the Continuous Flow method focuses minimally on retrospection. Team A follows Continuous Flow. Hence, the strategy of Retrospection was assigned a weight of zero in the OPS assessment. This strategy is linked to the above-mentioned principle. In effect, assigning a weight of zero for Retrospection nullified the strategy's impact on the higher-level principle of Frequent Reflection and Learning. On the other hand, the Expert scores considered that not holding retrospectives lowers the team's capability of implementing the principle: Frequent Reflection and Learning. Hence, the difference in the two sets of scores.

From the Pearson's correlation coefficient for the two sets of scores (see Table 6.6), we cannot conclude that the results are statistically significant. However, the trend lines are similar and, with the exception noted above, the differences in the two sets of scores are negligible.

***Table 6.6. Team A - Capability Assessment: Pearson Correlation Coefficient at the Principles Level***

| |
|---|
| **Pearson Correlation At the Principles Level – Capability Assessment** |
| **Computed OPS and Expert Values** |
| **r** = 0.21789, **N** = 8, **p-value** = 0.60421, two-tails |

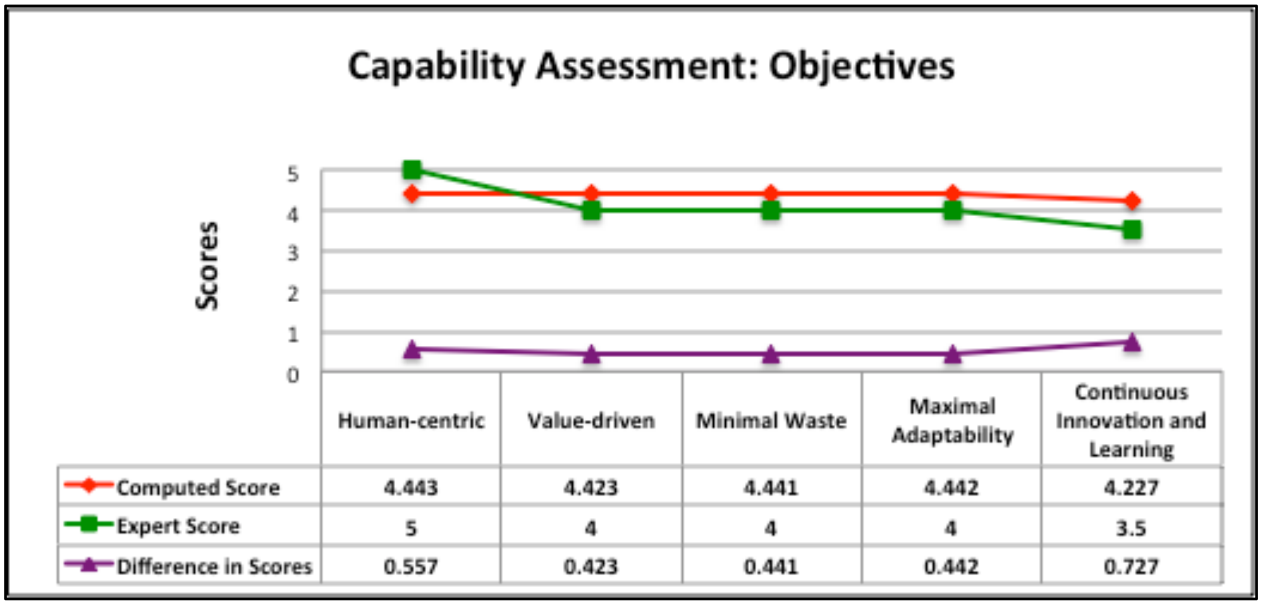
We conjecture that our inability to prove statistical significance stems from the fact that we initially assume equal weights, and therefore, as we propagate the scores up the indicator hierarchy, the *impact* of the individual scores is reduced.

***An Objectives Level Assessment***

In Figure 6.3, we see that both the Computed and Expert scores exhibit similar trends. For the objective Human-centric, the Expert scores are higher than the OPS score. As discussed previously, the Expert scores are implicitly weighted, whereas, in computing the OPS based scores we assume equal weights for all the linkages. Team A works as a cohesive unit. It is a mature agile team that works in a highly collaborative environment. The Experts have observed that this team is maximally capable of achieving the objective Human-centric. Consequently, the Expert score is higher than the Computed score.

From Figure 6.3, we also see that for the objective Continuous Innovation and Learning, there is a considerable difference in the two sets of scores. From the discussion about the Capability assessment at the principles and strategies levels, we know that the Experts view that fact Team A does not hold Retrospective meetings lowers the team's capability to support the principle Frequent Reflection. From the established linkages (see Appendix B), we know that there is a linkage path from Retrospection to Frequent Reflection and Learning and subsequently to Continuous Innovation and Learning. Given that

Team A adheres to the above-mentioned principle to a lesser extent, the Expert score for the objective under consideration is lower.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 4.443 | 4.423 | 4.441 | 4.442 | 4.227 |
| Expert Score | 5 | 4 | 4 | 4 | 3.5 |
| Difference in Scores | 0.557 | 0.423 | 0.441 | 0.442 | 0.727 |

***Figure 6.3. Team A - Capability Assessment: Objectives***

According to the Pearson Correlation factors shown in Table 6.7 below, the **r** and **p** values do not indicate any significant correlation.

***Table 6.7. Team A - Capability Assessment: Pearson Correlation Coefficient at the Objectives Level***

**Pearson Correlation At the Objectives Level – Capability Assessment**

**Computed OPS and Expert Values**

**r** = 0.63783,   **N** = 5,   **p-value** = 0.24693,   two-tails

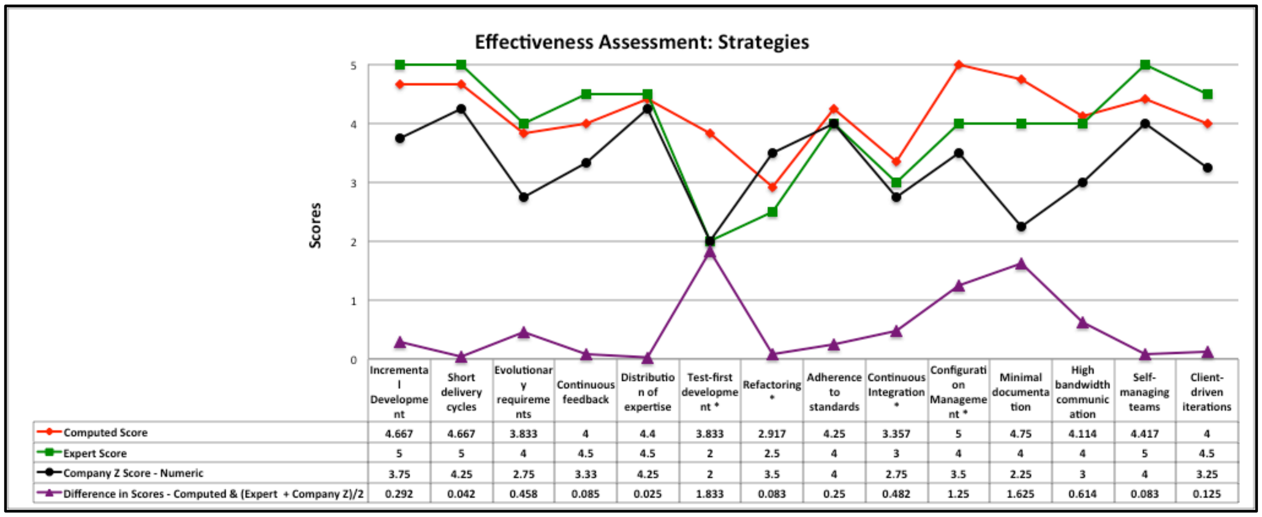
We conjecture that, as with the values assessed for principles, the significance and distinction of the contributing indicator values were minimized as each individual value was aggregated upward in the OPS hierarchy.  That "minimization effect" was not a factor in assigning the SME scores.

## Assessment: Effectiveness

Figures 6.4, 6.5, and 6.6 show the Effectiveness assessment results for the strategies, principles, and objectives. We see that the scores exhibit similar trends as evidenced by the above-mentioned figures. As mentioned previously, our effectiveness assessment is preliminary.

### *A Strategies Level Assessment*

At the strategies level, we see a substantial difference between the three sets of scores for Minimal Documentation (see Figure 6.4). The computed and expert scores align well. However, there is a sharp difference between these scores and the values assigned by the members of the development team. As described in Section 6.5, the reason can be attributed to the fact that the development team possess the capability to track their effectiveness over a considerable period of time. Also, our perceptions of maintaining minimal documentation may differ from those of the practitioners.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development * | Refactoring * | Adherence to standards | Continuous Integration * | Configuration Management * | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.667 | 4.667 | 3.833 | 4 | 4.4 | 3.833 | 2.917 | 4.25 | 3.357 | 5 | 4.75 | 4.114 | 4.417 | 4 |
| Expert Score | 5 | 5 | 4 | 4.5 | 4.5 | 2 | 2.5 | 4 | 3 | 4 | 4 | 4 | 5 | 4.5 |
| Company Z Score - Numeric | 3.75 | 4.25 | 2.75 | 3.33 | 4.25 | 2 | 3.5 | 4 | 2.75 | 3.5 | 2.25 | 3 | 4 | 3.25 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.292 | 0.042 | 0.458 | 0.085 | 0.025 | 1.833 | 0.083 | 0.25 | 0.482 | 1.25 | 1.625 | 0.614 | 0.083 | 0.125 |

*Figure 6.4. Team A - Effectiveness Assessment: Strategies*

In figure 6.4, we observe substantial differences in the scores for Test First Development. As mentioned previously, our effectiveness assessment is cursory especially with respect to the strategies Continuous Integration, Configuration Management, Refactoring and Test First Development.

The computed Pearson Correlation coefficient for the three sets of scores is shown in Table 6.8. We see that the results are statistically significant for the Computed and Expert Scores. The results are not statistically significant for the Computed and Company Z scores (see Table 6.8). We conjecture that this is due to the variability in the responses provided by Company Z personnel. Recall that Company Z personnel recorded their perceptions of their team's effectiveness with respect to the strategies on a five-point scale. The results shown as Company Z scores are the average of their responses. Practitioners have different perceptions of the extent to which their teams are effective in implementing the strategies. Therefore, there is some variability in their responses. Consequently, we are unable to show that the results are statistically significant with respect to the Computed and Company Z scores.

***Table 6.8. Team A - Effectiveness Assessment: Pearson Correlation Coefficient at the Strategies Level***

**Pearson Correlation At the Strategies Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

**r** = 0.67327,   **N** = 14,   **p-value** = 0.00831,   two-tails

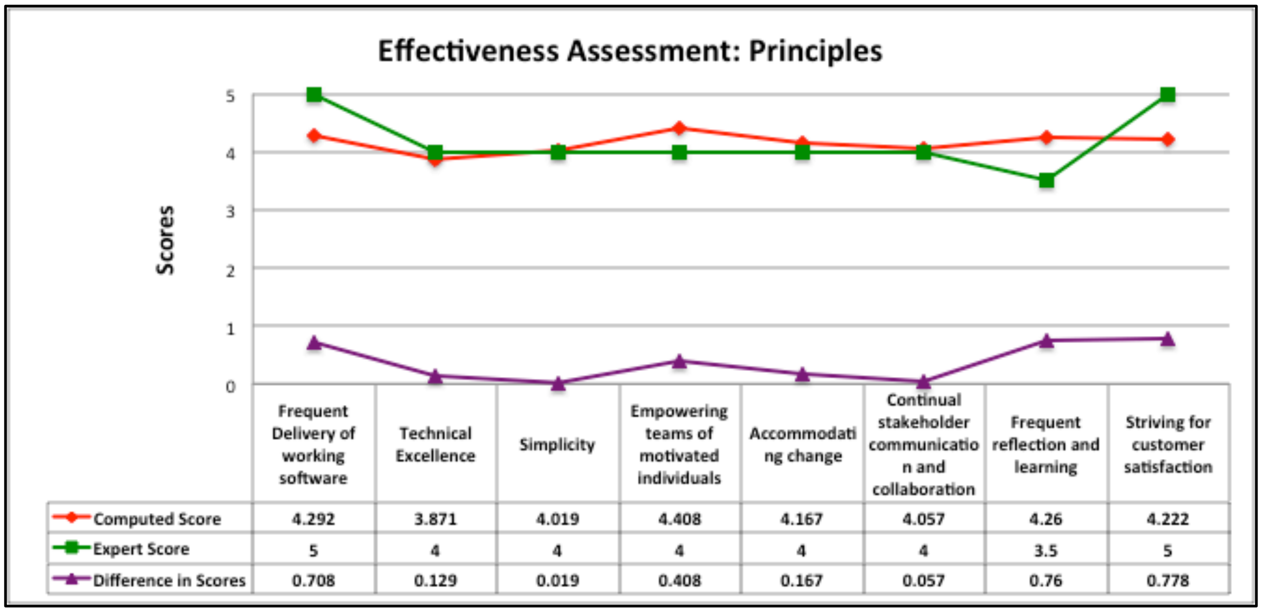
**Computed OPS and Company Z:**

**r** = 0.29618,   **N** = 14,   **p-value** = 0.30387,   two-tails

### *A Principles Level Assessment*

Figure 6.5 shows the effectiveness scores for the principles supported by Team A’s agile method. As mentioned previously, Team A does not hold retrospective meetings. From our observations, we see that this team does not take efforts to frequently reflect on their process and learn how to improve and innovate. Hence, the SME score for the principle ‘Frequent Reflection and Learning’ is lower. It is to be noted that the Expert scores are intrinsically weighted whereas the OPS scores consider all the linkages to be of equal weights. With respect to the other principles, we can see similar trends in the scores (Figure 6.5).

We also computed the Pearson correlation coefficient for the two sets of scores (see Table 6.9). Similar to the capability assessment results at the principles level, the results are not statistically significant. We surmise that we are unable to show statistical significance due to the lack of weights in the OPS Framework. Also, as we propagate the scores up the indicator hierarchies, the impact of the individual scores is minimized. We also conjecture that the results are not statistically significant at the principles level due to this minimization effect.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.292 | 3.871 | 4.019 | 4.408 | 4.167 | 4.057 | 4.26 | 4.222 |
| Expert Score | 5 | 4 | 4 | 4 | 4 | 4 | 3.5 | 5 |
| Difference in Scores | 0.708 | 0.129 | 0.019 | 0.408 | 0.167 | 0.057 | 0.76 | 0.778 |

*Figure 6.5. Team A - Effectiveness Assessment: Principles*

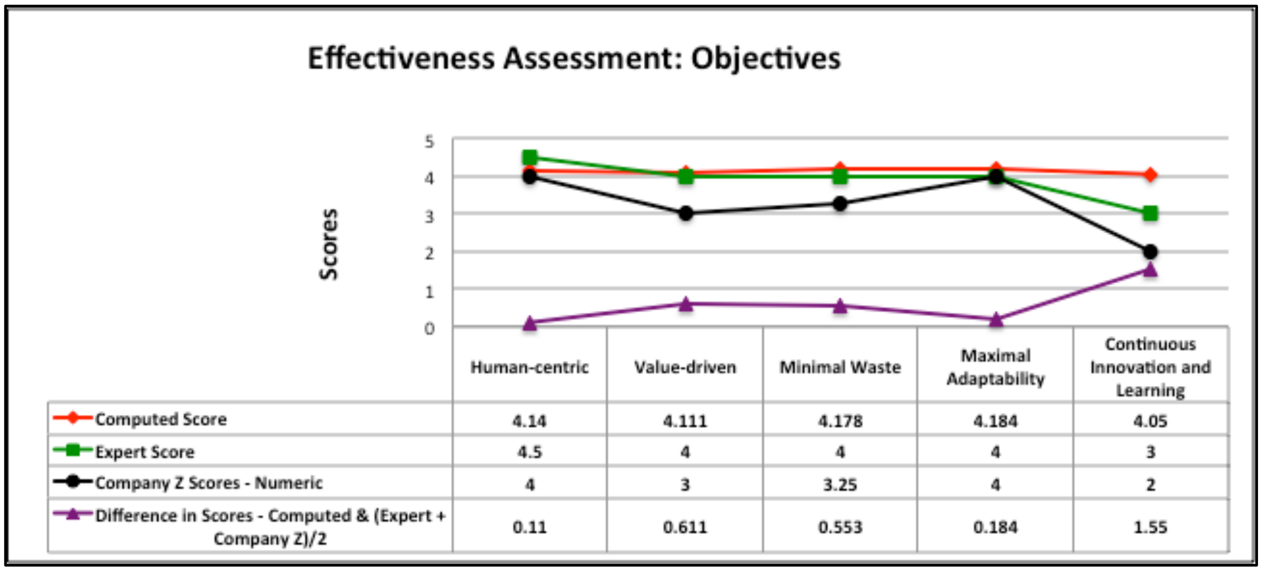
*Table 6.9. Team A - Effectiveness Assessment: Pearson Correlation Coefficient at the Principles level*

| Pearson Correlation At the Principles Level – Effectiveness Assessment (Preliminary) |
|---|
| **Computed OPS and Expert Values** |
| **r** = 0.22098,   **N** = 8,   **p-value** = 0.59896,   two-tails |

***An Objectives Level Assessment***

With respect to the objectives (see Figure 6.6), the differences in scores can be attributed to the differences in the perceptions of the experts and practitioners. Also, our effectiveness assessment is preliminary.

From Figure 6.6, we see that there is a considerable difference in the three sets of scores for ‘Continuous Innovation and Learning.’ It is to be noted that the scores follow similar trends. Team A personnel recognize that they do not innovate and improve their development process through the frequent examination and evaluation of past development activities. Hence, Company Z scores are lower for that objective. In Figure 6.6, all the three sets of scores exhibit similar trends.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 4.14 | 4.111 | 4.178 | 4.184 | 4.05 |
| Expert Score | 4.5 | 4 | 4 | 4 | 3 |
| Company Z Scores - Numeric | 4 | 3 | 3.25 | 4 | 2 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.11 | 0.611 | 0.553 | 0.184 | 1.55 |

***Figure 6.6. Team A - Effectiveness Assessment: Objectives***

The r and p values shown in Table 6.10 do not indicate any significant correlation among the three sets of scores. Recall that the linkages in the OPS Framework are not weighted and the results are preliminary. The Expert and Company Z scores on the other hand are inherently weighted. Hence, we are unable to show that the results are statistically significant.

***Table 6.10. Team A - Effectiveness Assessment: Pearson Correlation Coefficient at the Objectives Level***

| |
|---|
| **Pearson Correlation At the Objectives Level – Effectiveness Assessment (Preliminary)** |
| **Computed OPS and Expert:** |
| **r** = 0.71776, **N** = 5, **p-value** = 0.17217, two-tails |
| **Computed OPS and Company Z:** |
| **r** = 0.83921, **N** = 5, **p-value** = 0.0755, two-tails |

### *6.6.1.2 Team-specific Observations*

In this sub-section, we provide team-specific observations for Team A showcasing elements that are positive and those that require further examination.

**Positive:**

- In the recent past, Team A realized that a Scrum-like process was not meeting the team's needs with respect to frequently releasing software. Hence, they now follow the Continuous Flow approach, which has helped them release software more frequently. Currently, they have the "second fastest release cycle in the company."
- This team is mature. They understand their skills and accurately estimate the effort required for each story. Though they don't track the velocity, they usually complete about 6 points each week. They use "Anecdotal estimates" to guide their estimation process.
- In the Continuous Flow approach, the notion of constant velocity is inapplicable because the concept of iterations does not exist. However, a constant velocity of about 6 points per week is maintained.
- The Product Owner serves as the User Experience Designer.
- The developers support the Quality Engineer; story point estimates include the effort required for testing.
- The developers also help the Quality Engineer when they have completed the tasks that they had committed to for that release. They follow a "all hands on deck" approach before a release.
- Every member of this team understands the process they are to follow.
- They allow the requirements to evolve over time. Also, the team is capable of easily accommodating changes to requirements.
- Most of their releases have been successful; they have not had any rollbacks in the recent past.
- Dependencies between stories are discussed frequently.
- Members of Team A recognize the cost and value of each story.
- The team is self-managing.
- The Product Owner attends the daily stand-ups and is aware of the team's progress.
- Collective code ownership and knowledge sharing is promoted.
- The Product Owner maintains a continuous feedback loop with the customers.
- Team A is moving towards automated testing.
- Stories and releases are driven by anticipation of customer needs.

- The Product Owner and the development team negotiate when deciding on the stories to be developed.
- The team focuses on building the Minimum Viable Product.
- Before starting work on a project, the team determines the feasibility of that project.
- The developers estimate the effort required for each story and determine the priorities at the story level. Also, they constantly reassess the estimates and priorities of the features.

**Requires Scrutiny:**

- Team A does not hold retrospective meetings. In the past, the members did not find these meetings valuable. Hence, they stopped holding them. However, the Development Manager meets with each of his team members individually every week, and they use this time to reflect on their process, impediments, success factors, etc. Also, the daily stand-up meeting allows the team members to retrospect.
- At one time, the team practiced Continuous Integration. Some of the team members are now working on setting up an environment to resurrect this strategy.
- The developers do not have the opportunity to interact with the customers directly. However, their Product Owner and Development Manger share feedback from the customers and the users with the team.
- Occasionally, the meetings run over time.
- Test-first development is implemented only marginally.

#### *6.6.1.3 Summary*

We have examined the adequacy, capability, and effectiveness of Team A's agile method. The results are consistent with the practitioners' perceptions of their process. We first assessed the adequacy of the method. Team A follows Continuous Flow. By design, the method does not advocate certain strategies. Hence, certain objectives and principles are achieved to a lesser extent. More specifically, the composition of the agile method adopted by Team A, minimally supports the achievement of the objective 'Continuous Innovation and Learning.' The adequacy assessment results have been confirmed by Team A personnel.

To validate the OPS scores from a capability perspective, we computed the Pearson correlation coefficient for the objectives, principles, and strategies. The results are statistically significant at the strategies level. With respect to the principles, and objectives, we are unable to prove that the scores are statistically significant. We conjecture that this is due to the fact when the scores are propagated up the indicator hierarchy, the impact of the individual scores are minimized. From an effectiveness perspective, we see that the correlation between the OPS and Expert scores is statistically significant. In the other cases, we are unable to show statistical significance. We surmise that this can be attributed to the lack of weights in the OPS Framework and the minimization effect as the scores are propagated up the hierarchy. Also, we observed considerable variability in the responses provided by the Company Z personnel.

The computed OPS scores for capability and effectiveness for Team A fall in the 'Considerable' range. ***During the follow-up meeting with this team, the personnel confirmed our findings. They agreed with the assessment results and the team-specific observations that have also been included.*** The team recognizes that they "fall short" with respect to retrospection and "fine-tuning" their process.

## 6.6.2. OPS Based Assessment and Observations: Team B

This section summarizes the OPS based observations and assessment for Team B. This team is also involved in developing email applications. They follow Scrum. Table 6.11 below provides the team profile.

***Table 6.11. Profile - Team B***

| | |
|---|---|
| ***Team Size*** | 13 |
| ***Roles*** | Scrum Master (1) |
| | Product Owner (1) |
| | Technical Product Manager (1) |
| | Development Manager (1) |
| | Developers (6) |
| | Quality Engineer (1) |
| | Software Engineer in Test (1) |
| | Usability Engineer (1) |
| ***Development Process*** | Scrum |
| ***Tools used*** | VersionOne, SVN, TeamCity, NUnit, SpecFlow, Merlot, Redmine, Visual Studio, Trac |
| ***Velocity*** | The velocity is between 7 – 12 points every week. |
| ***Iteration length*** | 1 week. |

### *6.6.2.1 Team-specific Assessment*

Assessing the Capability and Effectiveness for Team B using the OPS Framework, we get the overall scores given below:

| **Capability: 4.257** | **Effectiveness: 3.941** |
|---|---|

Interpreting these scores using Table 6.2, we see that both values fall in the "***Considerable***" range (3.5 – 4.4). Hence, Team B is ***considerably*** (a) capable of implementing its adopted agile method and (b) effective in implementing that method.

#### ASSESSMENT: ADEQUACY

As a first step to assessing the capability and effectiveness of Team B, we examined the adequacy of their adopted agile method. Team B follows Scrum. Each iteration lasts one week. Examining the method, we recognize that Team B strives to achieve all the five objectives identified by the OPS Framework. In turn, it is expected that Team B support all the nine principles presented in the Framework. In Company Z as a whole, maintaining a 'Constant Development Pace' is of minimal value. Though the teams take historic velocity into consideration when estimating, they do not strive to maintain that velocity constant. Team B does not support the principle 'Constant Development Pace.' From the linkages in the OPS Framework, with the exception of 'Constant Velocity', Team B is expected to implement all the strategies provided in the OPS Framework. We observe that Team B uses all of those 16 strategies. We conclude that the agile method adopted by Team B is adequate with respect to achieving its stated objectives. Team B personnel have confirmed our findings with respect to adequacy.

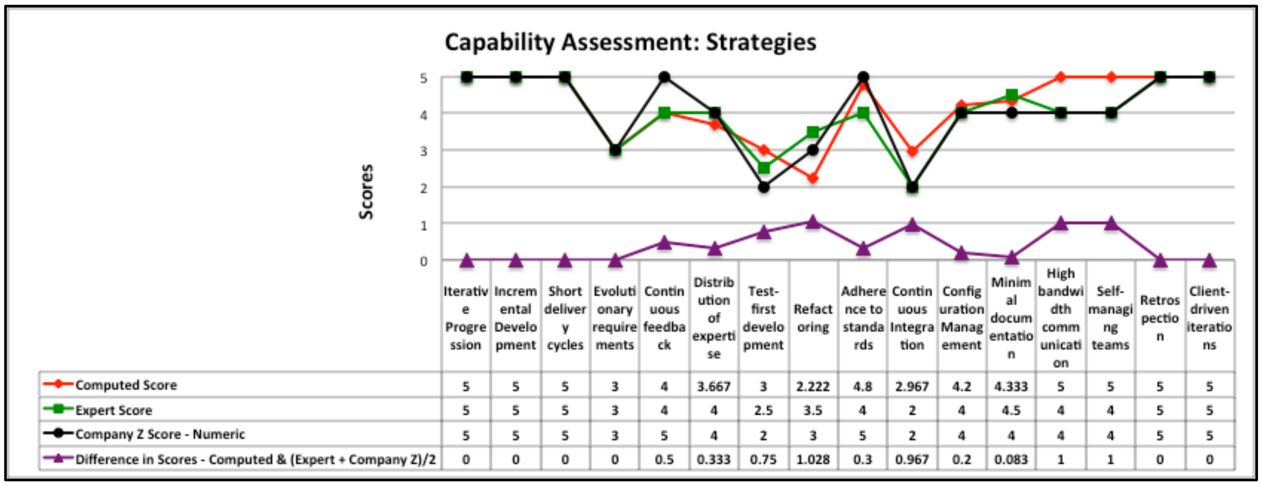
#### ASSESSMENT: CAPABILITY

Figures 6.7, 6.8, and 6.9 show the Capability Assessment results for the strategies, principles, and objectives respectively. From the above-mentioned figures, we see that the scores show similar trends.

***A Strategies Level Assessment***

In Figure 6.7, the strategy Continuous Integration has been assigned significantly lower scores. Though, Continuous Integration and Continuous Delivery (CICD) is a Company Z-wide initiative, Team B has the ability to implement Continuous Integration minimally. Sharing their code base with Teams C and D impedes the implementation of this strategy. Hence, the lower scores for that strategy (see Figure 6.7). In

Figure 6.7, all three sets of scores are considerably lower for the strategy 'Test-first Development.' In this team, it has been difficult for the practitioners to write tests first. Additionally, the developers work with legacy code. It is often impossible for this team to practice writing tests first when working with legacy code.



| | Iterative Progression | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development | Refactoring | Adherence to standards | Continuous Integration | Configuration Management | Minimal documentation | High bandwidth communication | Self-managing teams | Retrospection | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 5 | 5 | 5 | 3 | 4 | 3.667 | 3 | 2.222 | 4.8 | 2.967 | 4.2 | 4.333 | 5 | 5 | 5 | 5 |
| Expert Score | 5 | 5 | 5 | 3 | 4 | 4 | 2.5 | 3.5 | 4 | 2 | 4 | 4.5 | 4 | 4 | 5 | 5 |
| Company Z Score - Numeric | 5 | 5 | 5 | 3 | 5 | 4 | 2 | 3 | 5 | 2 | 4 | 4 | 4 | 4 | 5 | 5 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0 | 0 | 0 | 0 | 0.5 | 0.333 | 0.75 | 1.028 | 0.3 | 0.967 | 0.2 | 0.083 | 1 | 1 | 0 | 0 |

***Figure 6.7. Team B - Capability Assessment: Strategies***

Computing the Pearson correlation coefficient at the strategies level, we get the values shown below in Table 6.12:

***Table 6.12. Team B - Capability Assessment: Pearson Correlation Coefficient at the Strategies Level***

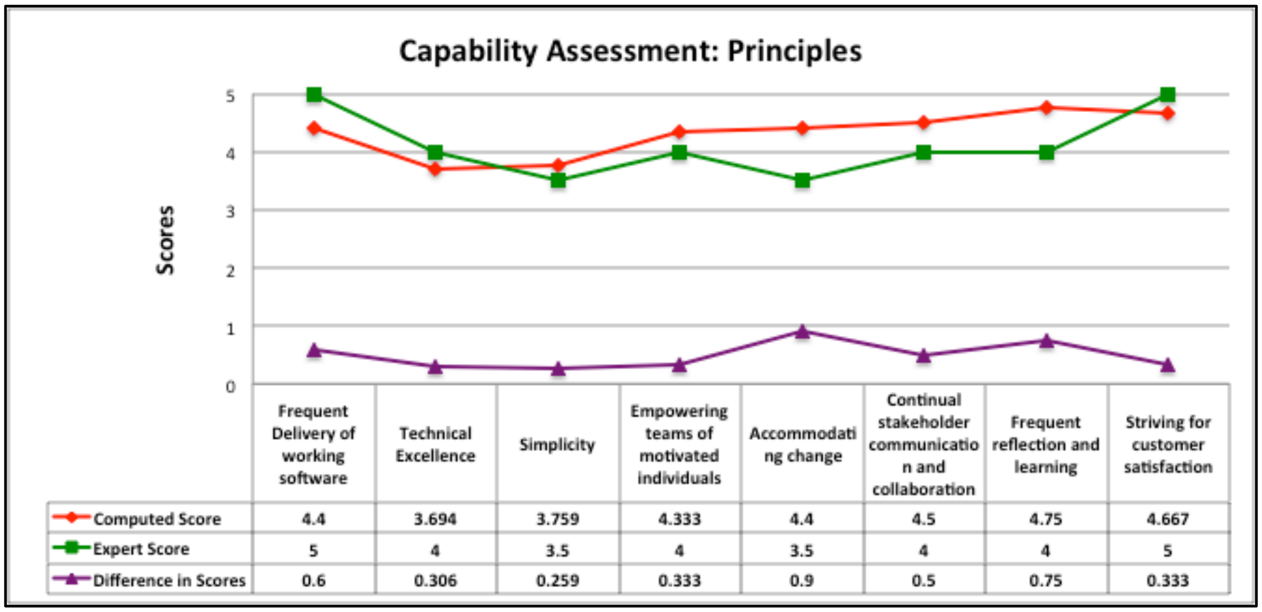
| **Pearson Correlation At the Strategies Level – Capability Assessment** |
|---|
| **Computed OPS and Expert:** |
| **r** = 0.80265, **N** = 16, **p-value** = 0.00018, two-tails |
| **Computed OPS and Company Z:** |
| **r** = 0.82315, **N** = 16, **p-value** = 0.00009, two-tails |

The correlation values show that the results obtained are ***statistically significant***. That is, the similarity in trends exhibited by the three sets of scores did not occur by random chance.

## *A Principles Level Assessment*

Figure 6.8 shows the capability assessment results for Team B at the principles level. From the figure, we see that the differences in the Computed and Expert scores are mostly negligible. With respect to the principle 'Accommodating Change', the Expert score is considerably lower than the Computed score (see

Figure 6.8). By design, Team B attempts to identify all the features and stories upfront. This considerably lowers the capability to accommodate change later in the development cycle. Also, the team members find it difficult to accommodate 'scope creep' and changes to features or stories. Hence, the Expert score for that principle is lower.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.4 | 3.694 | 3.759 | 4.333 | 4.4 | 4.5 | 4.75 | 4.667 |
| Expert Score | 5 | 4 | 3.5 | 4 | 3.5 | 4 | 4 | 5 |
| Difference in Scores | 0.6 | 0.306 | 0.259 | 0.333 | 0.9 | 0.5 | 0.75 | 0.333 |

***Figure 6.8. Team B - Capability Assessment: Principles***

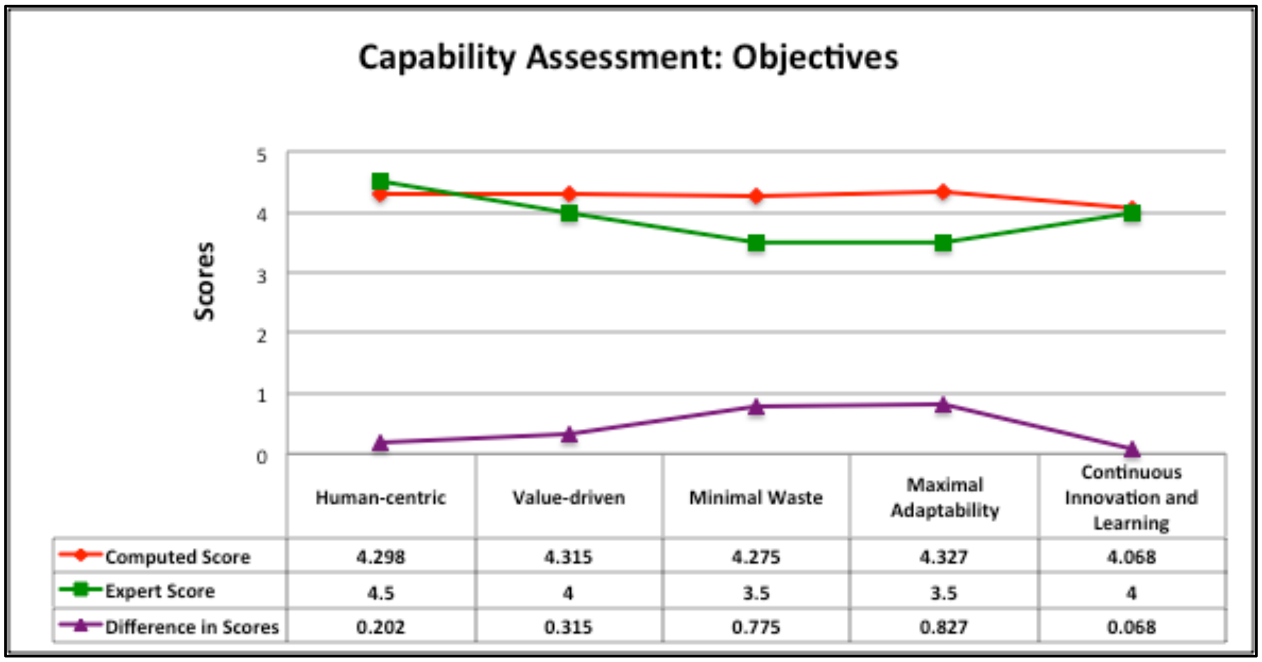
Computing the Pearson Correlation Coefficient at the principles level (see Table 6.13), we see that the results are not statistically significant. As discussed previously, we conjecture that this is due to the 'minimization effect' of the individual scores when propagated up the hierarchy of indicators.

***Table 6.13. Team B - Capability Assessment: Pearson Correlation Coefficient at the Principles Level***

| Pearson Correlation At the Principles Level – Capability Assessment |
|---|
| **Computed OPS and Expert Values** |
| **r** = 0.42581,   **N** = 8,   **p-value** = 0.29286,   two-tails |

***An Objectives Level Assessment***

In Figure 6.9, we observe that there are differences in the two sets of scores for the objectives Minimal Waste and Maximal Adaptability. Let us consider the objective Maximal Adaptability. Their Scrum-like process advocates the identification of all the stories upfront. Though they modify these stories during the development process, the requirements are not entirely allowed to evolve over time. This lowers the team's capability for achieving Maximal Adaptability. The expert's score weigh this aspect of the process more with respect to Maximal Adaptability. Hence, as shown in Figure 6.9, the Expert score is different from the computed OPS score.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 4.298 | 4.315 | 4.275 | 4.327 | 4.068 |
| Expert Score | 4.5 | 4 | 3.5 | 3.5 | 4 |
| Difference in Scores | 0.202 | 0.315 | 0.775 | 0.827 | 0.068 |

***Figure 6.9. Team B - Capability Assessment: Objectives***

The correlation at the objectives level is not statistically significant as indicated by the values shown below in Table 6.14:

***Table 6.14. Team B - Capability Assessment: Pearson Correlation Coefficient at the Objectives Level***

| **Pearson Correlation At the Objectives Level – Capability Assessment** |
|---|
| **Computed OPS and Expert Values** |
| **r** = -0.13209, **N** = 5, **p-value** = 0.83231, two-tails |

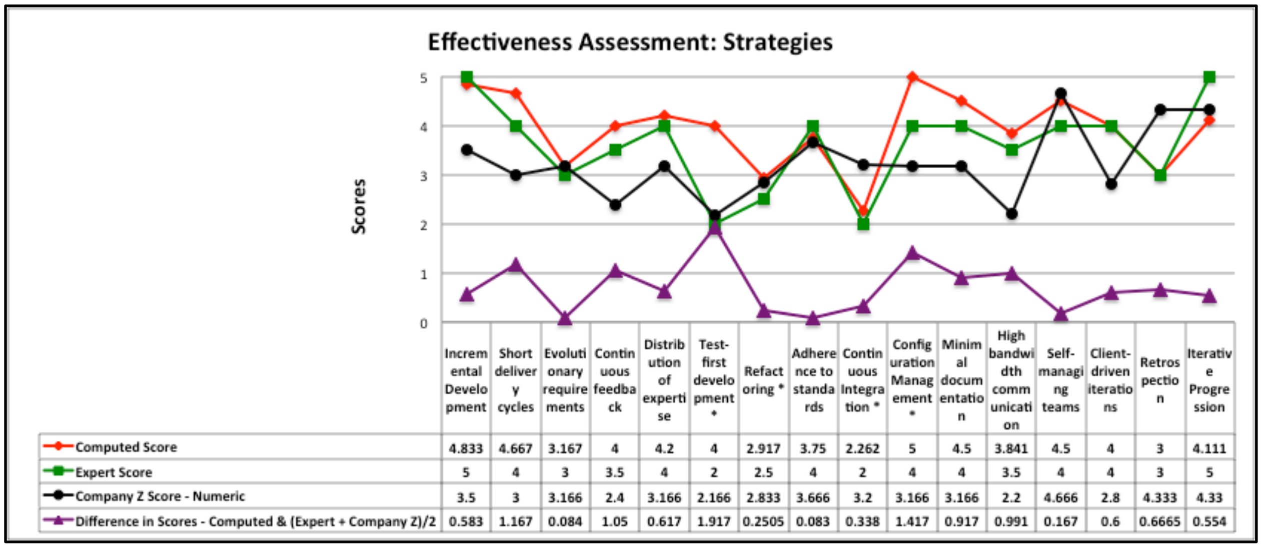
For Team B, the values indicate a negative correlation between the OPS and Expert scores. It is to be noted that the Expert scores are weighted and the OPS Framework assumes equal weights. Also, as the OPS scores are propagated to the higher levels in the assessment hierarchy, the impact of the individual scores is minimized. Hence, we are unable to show that the correlation values are statistically significant at the principles and objectives levels.

## ASSESSMENT: EFFECTIVENESS

Figures 6.10, 6.11, and 6.12 depict the Effectiveness Assessment results for Team B. From all three figures, we see that the scores mostly follow similar trends.

### *A Strategies Level Assessment*

In Figure 6.10, we see a difference in the Company Z scores and the other two scores for the strategies Incremental Development and High-bandwidth Communication. This can be attributed to the differences in perceptions of the Company Z practitioners and the experts. With respect to the strategies (Figure 6.10), we possess limited information to effect a complete assessment for the strategies Test First Development, Refactoring, Continuous Integration and Configuration Management. Hence, as shown in Figure 6.10, we see some differences in the numeric scores.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development * | Refactoring * | Adherence to standards | Continuous Integration * | Configuration Management * | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations | Retrospection | Iterative Progression |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.833 | 4.667 | 3.167 | 4 | 4.2 | 4 | 2.917 | 3.75 | 2.262 | 5 | 4.5 | 3.841 | 4.5 | 4 | 3 | 4.111 |
| Expert Score | 5 | 4 | 3 | 3.5 | 4 | 2 | 2.5 | 4 | 2 | 4 | 4 | 3.5 | 4 | 4 | 3 | 5 |
| Company Z Score - Numeric | 3.5 | 3 | 3.166 | 2.4 | 3.166 | 2.166 | 2.833 | 3.666 | 3.2 | 3.166 | 3.166 | 2.2 | 4.666 | 2.8 | 4.333 | 4.33 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.583 | 1.167 | 0.084 | 1.05 | 0.617 | 1.917 | 0.2505 | 0.083 | 0.338 | 1.417 | 0.917 | 0.991 | 0.167 | 0.6 | 0.6665 | 0.554 |

***Figure 6.10. Team B - Effectiveness Assessment: Strategies***

To validate the OPS scores, we computed the Pearson Correlation Coefficient at the strategies level for the (a) OPS and Expert, and (b) OPS and Company Z scores. The results are summarized in Table 6.15. We see that the correlation is significant for the OPS and the Expert scores. However, the correlation between the OPS and the Company Z scores is not statistically significant. As mentioned previously, we observed considerable variability in the responses provided by Company Z personnel. We conjecture that this variability can be attributed to the fact that the correlation between the OPS and Company Z scores is not significant.

*Table 6.15. Team B - Effectiveness Assessment: Pearson Correlation Coefficient at the Strategies Level*

**Pearson Correlation At the Strategies Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

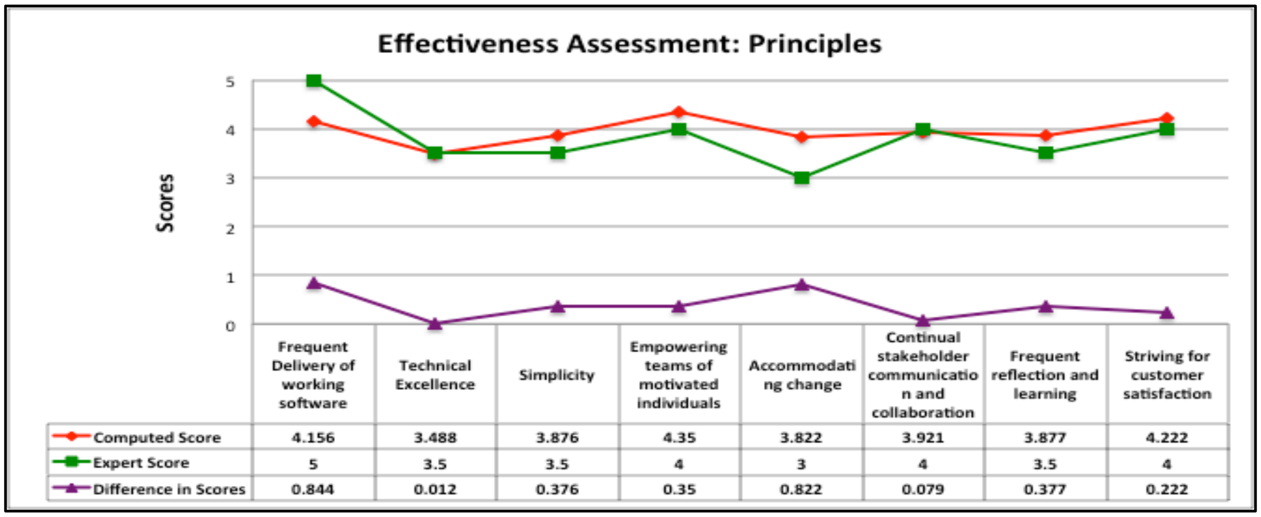
**r** = 0.73123, **N** = 16, **p-value** = 0.00129, two-tails

**Computed OPS and Company Z:**

**r** = 0.03566, **N** = 16, **p-value** = 0.89569, two-tails

## *A Principles Level Assessment*

With respect to the principles, we see that the scores follow similar trends (Figure 6.11). Since Team B attempts to 'freeze' the features and stories upfront, that team is minimally effective in accommodating change. Taking this observation into consideration, the Expert score for the principle 'Accommodating Change' is lower than the OPS-based score.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.156 | 3.488 | 3.876 | 4.35 | 3.822 | 3.921 | 3.877 | 4.222 |
| Expert Score | 5 | 3.5 | 3.5 | 4 | 3 | 4 | 3.5 | 4 |
| Difference in Scores | 0.844 | 0.012 | 0.376 | 0.35 | 0.822 | 0.079 | 0.377 | 0.222 |

*Figure 6.11. Team B - Effectiveness Assessment: Principles*

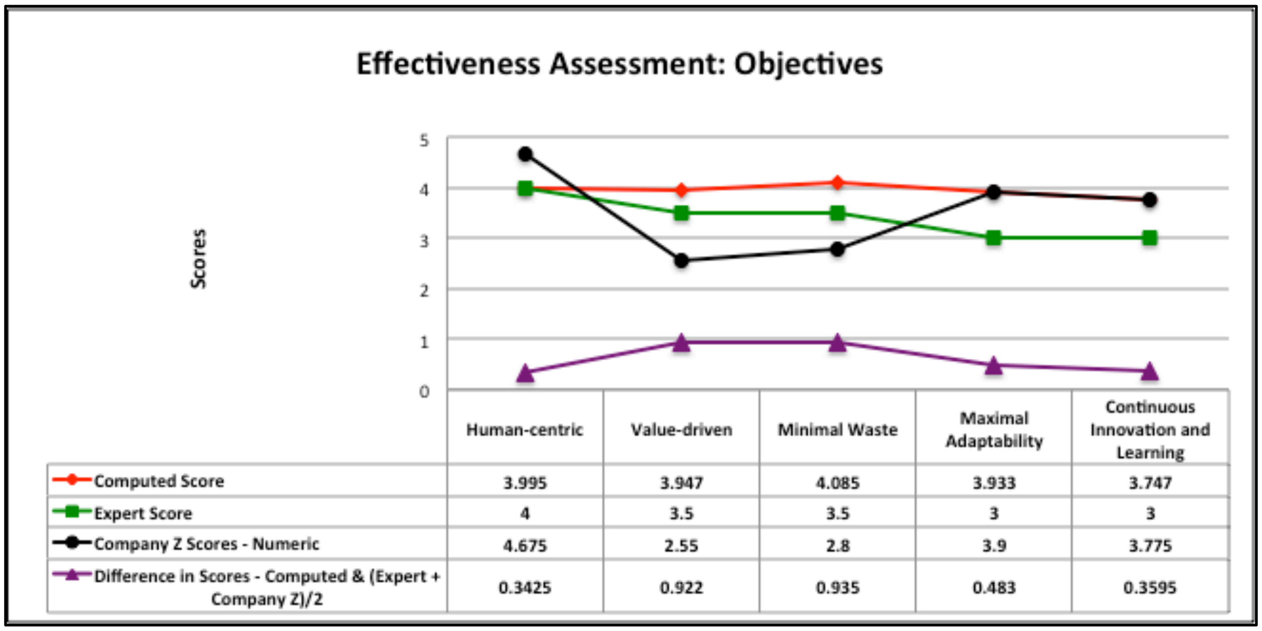
According the computed r and p values for the two sets of scores shown in Table 6.16, we see that the correlation is not statistically significant. We conjecture that we are unable to show statistical significance due the lack of weights in the OPS Framework. Also, our effectiveness assessment is preliminary.

***Table 6.16. Team B - Effectiveness Assessment: Pearson Correlation Coefficient at the Principles Level***

| |
|---|
| **Pearson Correlation At the Principles Level – Effectiveness Assessment (Preliminary)** |
| **Computed OPS and Expert Values** |
| **r** = 0.58414,   **N** = 8,   **p-value** = 0.12839,   two-tails |

***An Objectives Level Assessment***

In Figure 6.12, the differences in scores stem from the differences in perceptions of the experts and the practitioners. Also, our effectiveness assessment is preliminary. A longitudinal study has to be effected to provide a complete assessment of effectiveness.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| ◆ Computed Score | 3.995 | 3.947 | 4.085 | 3.933 | 3.747 |
| ■ Expert Score | 4 | 3.5 | 3.5 | 3 | 3 |
| ● Company Z Scores - Numeric | 4.675 | 2.55 | 2.8 | 3.9 | 3.775 |
| ▲ Difference in Scores - Computed & (Expert + Company Z)/2 | 0.3425 | 0.922 | 0.935 | 0.483 | 0.3595 |

***Figure 6.12. Team B - Effectiveness Assessment: Objectives***

Computing the correlation between the OPS and Expert, and the OPS and Company Z scores, we see that the results are not statistically significant at the objectives level (see Table 6.17). The OPS scores are not weighted whereas the Expert and Company Z scores are inherently weighted. This discrepancy contributes to the fact that the results are not statistically significant. It is to be noted that assessing effectiveness is a longitudinal process. Our study was implemented only for a short duration. Hence, our assessment of effectiveness has been cursory.

***Table 6.17. Team B - Effectiveness Assessment: Pearson Correlation Coefficient at the Objectives Level***

**Pearson Correlation At the Objectives Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

**r** = 0.61844, **N** = 5, **p-value** = 0.26614, two-tails

**Computed OPS and Company Z:**

**r** = -0.23225, **N** = 5, **p-value** = 0.70697, two-tails

### *6.6.2.2 Team-specific Observations*

Below are observations specific to Team B that outline the positive aspects of its people, process, product, project, and environment. Additionally, we also provide a set of observations whose implications should be reviewed for their impact.

**Positive:**

- The developers support the Quality Engineer and story point estimates include the effort required for testing.
- Retrospective meetings are held every 2-3 weeks.
- Team B schedules one release planning meeting to 'kick-off' of each project. During this meeting, they identify the scope of the project, the release milestones, and anticipated delivery dates. Though this requires considerable amount of work to be completed upfront, this helps the team understand the project vision, scope and requirements.
- The team is self-managing.
- They schedule "Engineering Weeks" to manage technical debt.
- The team now includes a User Experience Expert, a Software Development Engineer in Test, and a member of the Engineering team dedicated to work on projects specific to the B team.
- Team B has had minimal deployment rollbacks in the past two years.
- Collective code ownership and knowledge sharing is promoted.
- Developers work in pairs when necessary.
- The duration of the iterations is adjusted based on availability.

- The developers estimate the effort required for each story and determine the priorities at the story level.
- The internal customers (leadership, Product Owner) decide on the release milestones and establish the priorities for the features.
- Test plans are created before the developers start coding
- Refactoring is an expected activity. Stories dedicated to refactoring are added to the backlog.
- Team B usually completes the tasks that it had committed to during an iteration.
- Midway through an iteration, the team's progress is reviewed. Decisions regarding the scope of that iteration, story point estimates, and priorities are reassessed.
- The Product Owner maintains a continuous feedback loop with the customers.
- Team B is moving towards automated testing.

**Requires Scrutiny:**

- Each iteration lasts one week. Team B spends considerable amount of time in planning their sprint each week. This takes time away from development and testing. We understand the rationale for using one week iterations. However, the time spent on sprint planning each week seems excessive.
- The team members do include their velocity when estimating. However, they don't maintain a constant velocity.
- Some of the stories are assigned 0 points because these do not provide value in the form of deliverables. Examples of 0 point stories are research, engineering work, etc. These 0 point stories don't count towards the team's velocity. Though these stories involve effort, the team does not get credit for them. Hence, it is difficult to provide an accurate estimate of the team's velocity.
- Teams B, C and D share their codebase. This poses impediments to implementing Continuous Integration and Continuous Delivery.
- Occasionally, maximizing value to the customers is not considered a high priority goal. During one of the sprint planning meetings, the team had a discussion about "right thing for the customers vs right thing for the company". The developers discussed fixing an issue and said that it would add value to the customers. However, the Product Owner felt that fixing that issue would lower the company's revenue. He also said that in this case, "customer benefits are not a high priority".
- It has been difficult to get people to adopt TFD in this team.

- During most retrospectives, Development and Production issues with the software and the environments are listed as "things that did not go so well".
- The features/epics to be developed are identified upfront during the release planning meeting. Hence, the requirements are not allowed to evolve completely over the course of the development lifecycle.
- Accommodating changes to requirements is difficult.
- There seems to be a 'disconnect' between the Product Owner and the development team with respect to their understanding of the process.
- Retrospective goals are recorded but the team's efforts to achieve those goals are minimal.
- More often than not, meetings run over time. Also, the meetings are rescheduled frequently.
- The development team is removed from the external customers.

#### *6.6.2.3 Summary*

Guided by the OPS Framework, we first examined the adequacy of Team B's development approach. The method was found adequate with respect to achieving the stated objectives. With the exception of the principle "Constant Development Pace' and the strategy 'Constant Velocity', the agile method under consideration is composed of all the expected principles and strategies that are necessary to achieve the stated objectives. During the follow-up meeting with Team B, the practitioners concurred with our assessment of adequacy of their process.

All three sets of scores – Computed, Expert, and Company Z exhibit similar trends from a capability and an effectiveness perspective. With respect to capability, the results are statistically significant at the strategies level. From an effectiveness perspective, the correlation between the OPS and Company Z scores is significant at the strategies level. At the principles and objectives levels, the scores are not statistically significant because of the minimization of the impact of individual scores as they are propagated up the hierarchy. Also, our assessment of effectiveness is preliminary.

From the assessment results, Team B is 'considerably' (a) capable of implementing its adopted agile method and (b) effective in producing expected results. ***Team B personnel confirmed our findings (OPS based assessment and the team-specific observations).*** The follow-up meetings were scheduled three months after the conclusion of the onsite examination of the four teams. In that three-month period, Team B had taken steps to address some of our observations listed in sub-section 6.6.2.2. For example, they no longer estimate stories to be zero points. They have recognized that including such zero point stories does not allow them to accurately determine their velocity. Additionally, the team realizes that they are not achieving the retrospective goals. To address this issue, they are planning to provide greater visibility to these goals.

## 6.6.3. OPS Based Assessment and Observations: Team C

Table 6.18 below outlines Team C’s profile. This team follows a Scrum-like process. The processes of Teams B and C are very similar. Since Team C is relatively newly formed, they are still working towards establishing their process. This team is primarily involved in building financial applications that are intended for internal use.

***Table 6.18. Profile - Team C***

| | |
|---|---|
| ***Team Size*** | 9 |
| ***Roles*** | Scrum Master (1)<br>Product Owner (1)<br>Development Manager (1)<br>Developers (5)<br>Quality Engineer (1) |
| ***Development Process*** | Scrum |
| ***Tools used*** | VersionOne, SVN, TeamCity, NUnit, SpecFlow, Merlot, Redmine, Visual Studio, Trac |
| ***Velocity*** | The members of this team have only been working together for about three months. They are still trying to establish their velocity.<br>Currently, they have been able to deliver about 30 story points in two weeks. |
| ***Iteration length*** | 2 weeks. |

### *6.6.3.1 Team-specific Assessment*

Based on our OPS observations and assessments, the overall Capability and Effectiveness scores for Team C are:

| **Capability: 4.227** | **Effectiveness: 3.886** |
|---|---|

Using Table 6.2 to interpret these scores, both values fall in the "***Considerable***" range (3.5 – 4.4). Hence, Team C is ***considerably*** (a) capable of implementing its adopted agile method and (b) effective in implementing that method.

#### **ASSESSMENT: ADEQUACY**

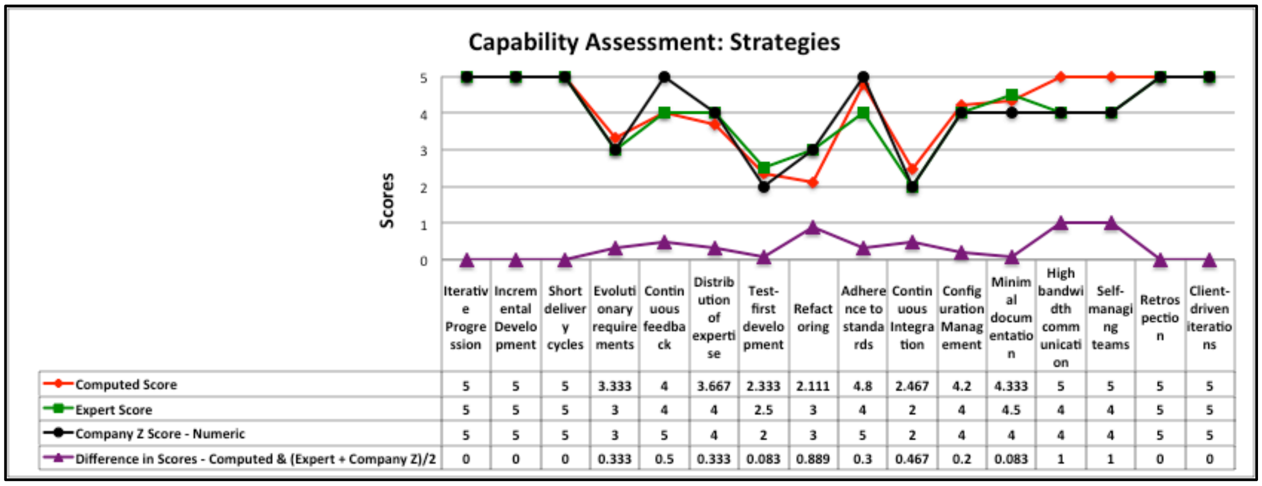
Like Team B, Team C also follows a Scrum-like approach. Each iteration lasts for two weeks. At the end of an iteration, a software increment is deployed. Examining this approach adopted by Team C, we recognize that the team strives to achieve all the five objectives identified by the OPS Framework. Guided by the linkages in the Framework, we know that to achieve those five objectives, all the nine identified principles have to be adhered to. As mentioned previously, in Company Z, maintaining a 'Constant Development Pace' is not a priority. Hence, Team C does not support that principle. Once again, from the OPS Framework, we know that with the exception of 'Constant Velocity', Team C should implement the remaining 16 strategies. From our observations, we see that Team C implements all those strategies. To summarize, Team C's adopted agile method is adequate with respect to achieving its stated objectives. During the follow-up meeting with Team C, the team members confirmed our adequacy assessment results.

#### **ASSESSMENT: CAPABILITY**

Figures 6.13, 6.14, and 6.15 show the Capability Assessment results for the strategies, principles, and objectives respectively. As shown in these figures, the scores follow similar trends.

#### ***A Strategies Level Assessment***

At the strategies level, we present three sets of scores – Computed OPS, Expert, and Company Z. As shown in Figure 6.13, all three sets of scores show similar trends. This team identifies all the requirements upfront during their release planning meeting. By the design of their process, the features and stories are minimally allowed to evolve over time. Hence, the scores for the strategy Evolutionary Requirements are lower (see Figure 6.13). Support for Test-first Development and Continuous Integration is marginal. From Figure 6.13, we observe that all three sets of scores are lower for those strategies as well.



| | Iterative Progression | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development | Refactoring | Adherence to standards | Continuous Integration | Configuration Management | Minimal documentation | High bandwidth communication | Self-managing teams | Retrospection | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 5 | 5 | 5 | 3.333 | 4 | 3.667 | 2.333 | 2.111 | 4.8 | 2.467 | 4.2 | 4.333 | 5 | 5 | 5 | 5 |
| Expert Score | 5 | 5 | 5 | 3 | 4 | 4 | 2.5 | 3 | 4 | 2 | 4 | 4.5 | 4 | 4 | 5 | 5 |
| Company Z Score - Numeric | 5 | 5 | 5 | 3 | 5 | 4 | 2 | 3 | 5 | 2 | 4 | 4 | 4 | 4 | 5 | 5 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0 | 0 | 0 | 0.333 | 0.5 | 0.333 | 0.083 | 0.889 | 0.3 | 0.467 | 0.2 | 0.083 | 1 | 1 | 0 | 0 |

*Figure 6.13. Team C - Capability Assessment: Strategies*

To validate the computed OPS values at the strategies level, we have determined the Pearson correlation coefficient between the sets of values for (a) the Computed OPS and SMEs, and (b) the Computed OPS and Company Z. The results are shown in Table 6.19.

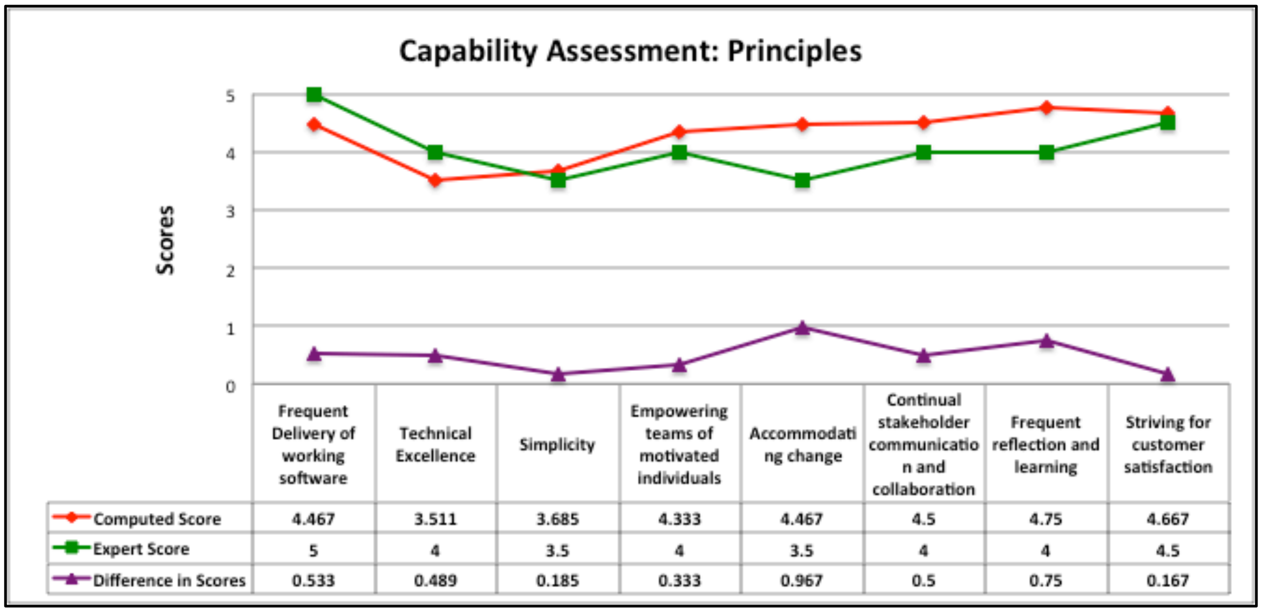
*Table 6.19. Team C - Capability Assessment: Pearson Correlation Coefficient at the Strategies Level*

| **Pearson Correlation At the Strategies Level** |
|---|
| **Computed OPS and Expert:** |
| **r** = 0.88427, **N** = 16, **p-value** = 0.00001, two-tails |
| **Computed OPS and Company Z:** |
| **r** = 0.86828, **N** = 16, **p-value** = 0.00001, two-tails |

From the values shown above, we see that the results are statistically significant at the strategies level. That is, the similarity in trends shown by the Computed, Expert, and Company Z scores is not by random chance.

***A Principles Level Assessment***

As shown in Figure 6.14, the Computed and Expert scores predominantly follow similar trends. Recall that all the linkages in the OPS Framework have been assigned ***equal weights***. In Figure 6.14, we observe that there is a substantial difference in the two sets of scores for the principle: Accommodating Change. With Team C, we have observed that the development team is minimally receptive to accommodating changes to the requirements. The Expert score weighs this aspect more with respect to the principle Accommodating Change. Hence, the difference in the two sets of scores.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.467 | 3.511 | 3.685 | 4.333 | 4.467 | 4.5 | 4.75 | 4.667 |
| Expert Score | 5 | 4 | 3.5 | 4 | 3.5 | 4 | 4 | 4.5 |
| Difference in Scores | 0.533 | 0.489 | 0.185 | 0.333 | 0.967 | 0.5 | 0.75 | 0.167 |

***Figure 6.14. Team C - Capability Assessment: Principles***

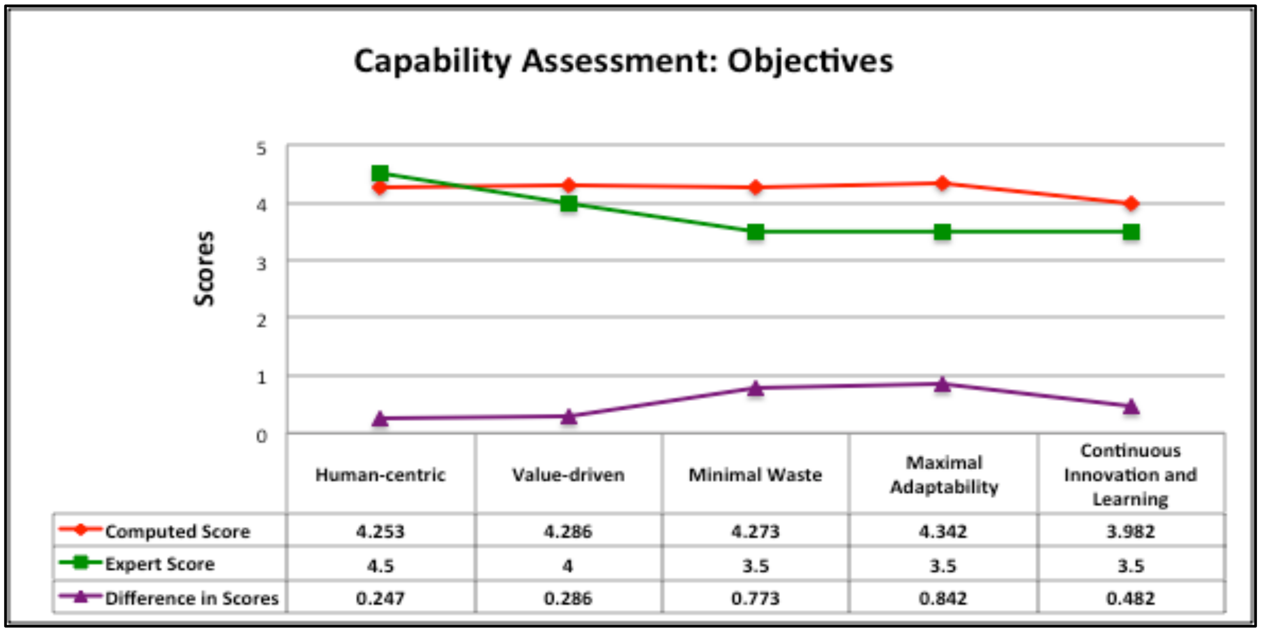
From the Pearson's correlation coefficient for the two sets of scores, we cannot conclude that the results are statistically significant (see Table 6.20). However, the trend lines are similar and, with the exception noted above, the differences in the two sets of scores are negligible. We surmise that as we propagate the scores to the higher levels of the assessment hierarchy, the impact of the individual scores are minimized.

***Table 6.20. Team C - Capability Assessment: Pearson Correlation Coefficient at the Principles Level***

| |
|---|
| **Pearson Correlation At the Principles Level** |
| **Computed OPS and Expert Values** |
| **r** = 0.36664, **N** = 8, **p-value** = 0.37167, two-tails |

***An Objectives Level Assessment***

Figure 6.15 depicts the assessment results with respect to the objectives. Similar to the results obtained for the principles, we observe differences in the Computed and Expert scores. In particular, we see considerable differences in the two sets of scores for the objectives 'Maximal Adaptability' and 'Minimal Waste' (see Figure 6.15). Similar to Team B, Team C also identifies all the features and stories to be developed upfront. Also, the team members find it difficult to accommodate changes to requirements later in the development lifecycle. This lowers Team C's capability to achieve 'Maximal Adaptability'. With respect to 'Minimal Waste', the teams focus on building "more than what is necessary." Consequently, the capability to achieve that objective is lessened.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 4.253 | 4.286 | 4.273 | 4.342 | 3.982 |
| Expert Score | 4.5 | 4 | 3.5 | 3.5 | 3.5 |
| Difference in Scores | 0.247 | 0.286 | 0.773 | 0.842 | 0.482 |

***Figure 6.15. Team C - Capability Assessment: Objectives***

According to the Pearson Correlation factors shown in Table 6.21, the **r** and **p** values do not indicate any significant correlation. We conjecture that, as with the values assessed for principles, the significance and distinction of the contributing indicator values were minimized as each individual value was aggregated upward in the OPS hierarchy. That "minimization effect" was not a factor in assigning the SME scores.

***Table 6.21. Team C - Capability Assessment: Pearson Correlation Coefficient at the Objectives Level***

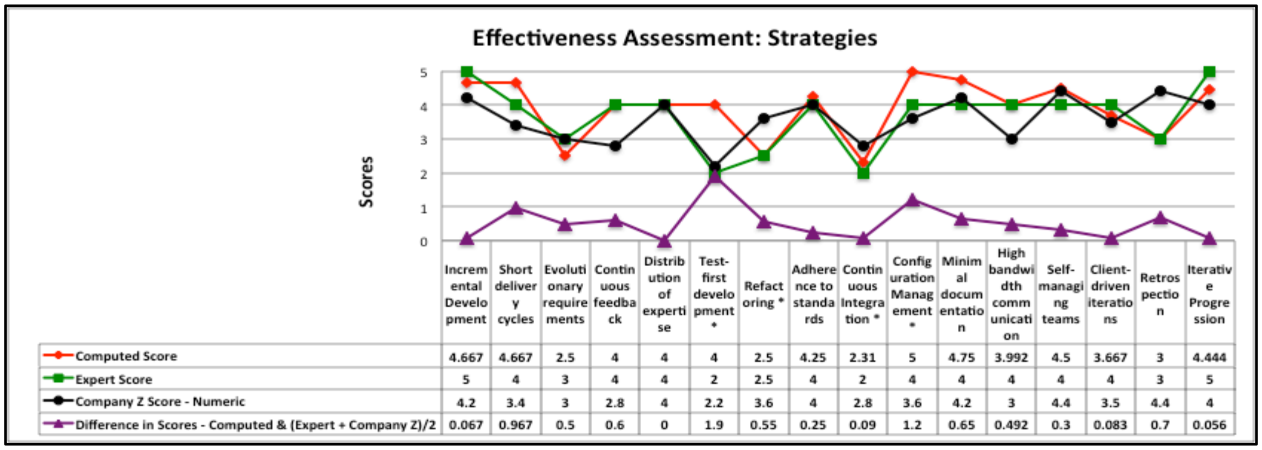
| |
|---|
| **Pearson Correlation At the Objectives Level** |
| **Computed OPS and Expert Values** |
| **r** = 0.21885, **N** = 5, **p-value** = 0.72359, two-tails |

## ASSESSMENT: EFFECTIVENESS

Figures 6.16, 6.17, and 6.18 depict the Effectiveness Assessment results for Team C. From all three figures, it is evident that the scores predominantly exhibit similar trends.

### *A Strategies Level Assessment*

With respect to the strategies (Figure 6.16), we see a difference between the Company Z scores and the other two scores for Retrospection. The mechanics of implementation for Retrospection include the identification and achievement of retrospective goals. During the retrospective meetings, Team C identifies retrospective goals, but takes minimal efforts to achieve them. Based on this observation, the OPS and the expert scores regarding the effectiveness of Retrospection are lower than those assigned by the practitioners.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development * | Refactoring * | Adherence to standards | Continuous Integration * | Configuration Management * | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations | Retrospection | Iterative Progression |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.667 | 4.667 | 2.5 | 4 | 4 | 4 | 2.5 | 4.25 | 2.31 | 5 | 4.75 | 3.992 | 4.5 | 3.667 | 3 | 4.444 |
| Expert Score | 5 | 4 | 3 | 4 | 4 | 2 | 2.5 | 4 | 2 | 4 | 4 | 4 | 4 | 4 | 3 | 5 |
| Company Z Score - Numeric | 4.2 | 3.4 | 3 | 2.8 | 4 | 2.2 | 3.6 | 4 | 2.8 | 3.6 | 4.2 | 3 | 4.4 | 3.5 | 4.4 | 4 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.067 | 0.967 | 0.5 | 0.6 | 0 | 1.9 | 0.55 | 0.25 | 0.09 | 1.2 | 0.65 | 0.492 | 0.3 | 0.083 | 0.7 | 0.056 |

*Figure 6.16. Team C - Effectiveness Assessment: Strategies*

Computing the Pearson Correlation Coefficient for the OPS scores and the other two sets of scores (Table 6.22), we see that the results are statistically significant with respect to the OPS and Expert scores. The correlation between the OPS and Company Z scores is not significant. Company Z personnel have divergent views about the extent to which they are effective in implementing the strategies. Hence, there is considerable variability in the collected data. We surmise that the variability contributes to the fact that we are unable to show statistical significance for the OPS and Company Z scores.

*Table 6.22. Team C - Effectiveness Assessment: Pearson Correlation Coefficient at the Strategies Level*

**Pearson Correlation At the Strategies Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

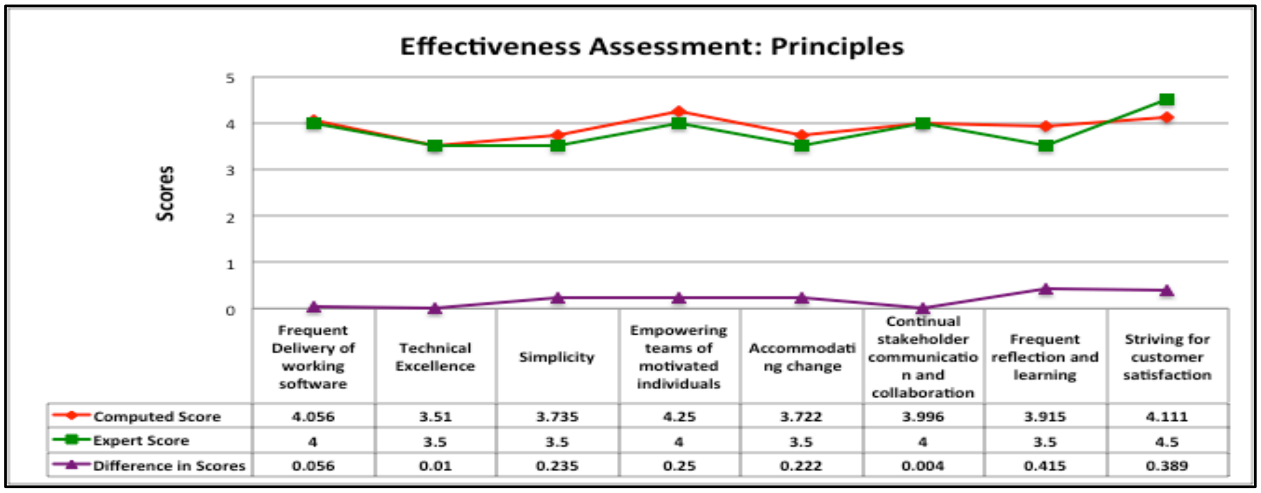
**r** = 0.73301, **N** = 16, **p-value** = 0.00124, two-tails

**Computed OPS and Company Z:**

**r** = 0.33176, **N** = 16, **p-value** = 0.20937, two-tails

## *A Principles Level Assessment*

Figure 6.17 shows the effectiveness assessment results with respect to the principles. From the figure, we see that the Computed and Expert scores show similar trends. Also, the differences in the two sets of scores are negligible. As discussed previously, the team does not strive to achieve the retrospective goals. This lowers the team's effectiveness with respect to adhering to the principle 'Frequent Reflection and Learning.'



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| ◆ Computed Score | 4.056 | 3.51 | 3.735 | 4.25 | 3.722 | 3.996 | 3.915 | 4.111 |
| ■ Expert Score | 4 | 3.5 | 3.5 | 4 | 3.5 | 4 | 3.5 | 4.5 |
| ▲ Difference in Scores | 0.056 | 0.01 | 0.235 | 0.25 | 0.222 | 0.004 | 0.415 | 0.389 |

*Figure 6.17. Team C - Effectiveness Assessment: Principles*

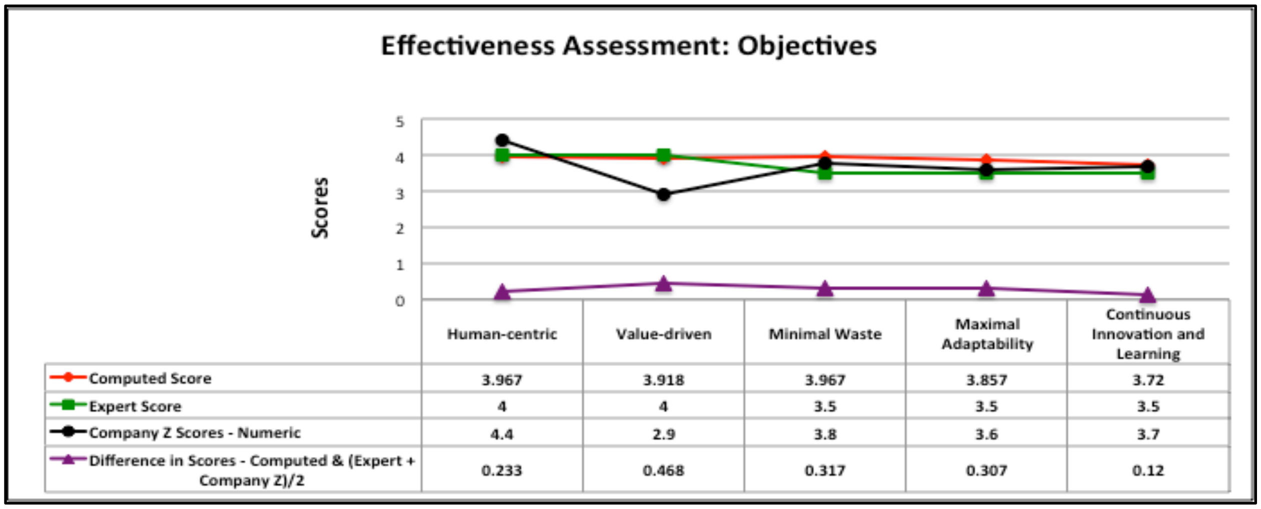
To validate the OPS scores, we computed the Pearson Correlation Coefficient for the OPS and the Expert scores. From Table 6.23, we see that the results are not statistically significant. As discussed previously, the OPS scores are not weighted. However, the Expert scores are based on perceptions and are inherently weighted. Also, as scores are propagated up the indicator hierarchies, we see that the impact of individual scores is minimized. We conjecture that we are unable to show statistical significance at the principles level from an effectiveness perspective due to the lack of weights in the OPS Framework and the minimization effect.

***Table 6.23. Team C - Effectiveness Assessment: Pearson Correlation Coefficient at the Principles Level***

| **Pearson Correlation At the Principles Level – Effectiveness Assessment (Preliminary)** |
|---|
| **Computed OPS and Expert Values** |
| **r** = 0.76467, **N** = 8, **p-value** = 0.0271, two-tails |

***An Objectives Level Assessment***

The effectiveness assessment with respect to the objectives is summarized in Figure 6.18. In that figure, we observe differences in the there sets of scores for the objective Value-driven. The Computed and Expert scores align but the Company Z scores are considerably lower. The variations stem from the differences in perceptions of the experts and the practitioners. It is also to be noted that our effectiveness assessment is preliminary.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 3.967 | 3.918 | 3.967 | 3.857 | 3.72 |
| Expert Score | 4 | 4 | 3.5 | 3.5 | 3.5 |
| Company Z Scores - Numeric | 4.4 | 2.9 | 3.8 | 3.6 | 3.7 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.233 | 0.468 | 0.317 | 0.307 | 0.12 |

***Figure 6.18. Team C - Effectiveness Assessment: Objectives***

From Table 6.24, we see that there is no significant correlation between the OPS and Expert, and the OPS and Company Z scores. As mentioned previously, we surmise that this is due to (a) the lack of weights in the OPS Framework, (b) the minimization effect as the scores are propagated up the indicator hierarchies, and (c) variability in the responses provided by the Company Z personnel. Additionally, our assessment of effectiveness is preliminary.

***Table 6.24. Team C - Effectiveness Assessment: Pearson Correlation Coefficient at the Objectives Level***

**Pearson Correlation At the Objectives Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

**r** = 0.50189, **N** = 5, **p-value** = 0.38892, two-tails

**Computed OPS and Company Z:**

**r** = 0.19041, **N** = 5, **p-value** = 0.75903, two-tails

### *6.6.3.2 Team-specific Observations*

In this sub-section, we list observations specific to Team C. The effects of some of these observations are positive and others should be reviewed for their impact.

**Positive:**

- The developers support the Quality Engineer; story point estimates include the effort required for testing.
- Retrospective meetings are held every 2 weeks.
- Team C schedules one release planning meeting to 'kick-off' each project. During this meeting, they identify the scope of the project, the release milestones, and anticipated delivery dates. Though this requires considerable amount of work to be completed upfront, this helps the team understand the project vision, scope and requirements.
- The team is self-managing.
- They add stories to manage technical debt.
- Collective code ownership and knowledge sharing is promoted.
- Developers work in pairs when necessary.
- The duration of the iterations is adjusted based on personnel availability.
- The developers estimate the effort required for each story and determine the priorities at the story level.
- The internal customers (leadership, Product Owner) decide on the release milestones and establish the priorities for the features.

- Test plans are created before the developers start coding.
- Refactoring is an expected activity. Stories dedicated to refactoring are added to the backlog.
- Team C mostly completes the tasks that it commits to during an iteration.
- Midway through an iteration, the team's progress is reviewed. Decisions regarding the scope of that iteration, story point estimates, and priorities are reassessed.
- The Product Owner maintains a continuous feedback loop with the customers.
- Though Team C is newly formed, the team members work very well together.
- The Product Owner attends the daily stand-ups, and is aware of the team's progress. Moreover, the Product Owner regularly shares feedback and updates from the customers and leadership with the team.
- Developers know their skill sets and ensure that they are not assigned tasks that they are incapable of finishing.
- When hard deadlines for completing projects is set, the Product Owner and the development team discuss and appropriately adjust the scope for the project.

**Requires Scrutiny:**

- Since they are a new team, they are still trying to establish their velocity.
- Teams C, B and D share their codebase. This poses impediments to implementing Continuous Integration and Continuous Delivery.
- Occasionally, maximizing value is constrained by time and other factors.
- It has been difficult to get people to adopt TFD in this team.
- During most retrospectives, Development and Production issues with the software and the environments are listed as "things that did not go so well". The team members spend considerable amounts of time fixing the environment issues.
- The features/epics to be developed are identified upfront during the release planning meeting. Hence, the requirements are not completely allowed to evolve over the course of the development lifecycle.
- Accommodating changes to requirements is difficult. The team finds it challenging to manage scope creep.
- Retrospective goals are recorded, but the team's efforts to achieve those goals are minimal.

- More often than not, meetings run over time. Also, the meetings are rescheduled frequently.
- The development team is removed from the external customers.
- Some of the team members arrive late for meetings.
- The team has faced issues when communicating and collaborating with the teams in San Antonio.
- Occasionally, the developers do not hand over the stories to the Quality Engineer until later during the iteration. This causes a bottleneck.

### *6.6.3.3 Summary*

The Scrum-like approach adopted by Team C is adequate with respect to achieving its stated objectives. Team C's process is almost identical to the one adopted by Team B. With the exception of the principle 'Constant Development Pace' and the strategy 'Constant Velocity', Team C's method is composed of all the expected principles and strategies (derived from the OPS Framework) necessary to achieve its stated objectives. Team C's personnel have confirmed our adequacy assessment of their adopted agile method.

As discussed in Section 6.6.3.2, the overall capability and effectiveness scores for Team C fall in the considerable range. With respect to capability, the results are also statistically significant at the strategies level. From an effectiveness perspective, the correlation is significant for the OPS and Expert scores at the strategies level. Due to the 'minimization effect', the scores obtained for the principles and objectives are not statistically significant. Also, there is considerable variability in the data gathered from Company Z personnel with respect to effectiveness.

***Team C personnel have confirmed the OPS-based assessments and team-specific observations.*** The team recognized that identifying all the features upfront is lowering their ability to accommodate change. On the other hand, such an upfront effort helps them understand the scope of the project. Currently, the team is working to find a balance. For a relatively newly formed team, Team C works very well together and has made significant progress is achieving their goals.

## 6.6.4. OPS Based Assessment and Observations: Team D

This section summarizes the OPS based observations and assessment for Team D. This team has recently moved away from a Scrum-like process. Scrum was judged to be process-heavy. They now follow Kanban. Like Teams A and B, members of this team are also involved in building email applications. Table 6.25 below provides an overview of this team's profile.

***Table 6.25. Profile - Team D***

| | |
|---|---|
| ***Team Size*** | |
| | 11 |
| ***Roles*** | |
| | Scrum Master (1) |
| | Product Owner (1) |
| | Technical Product Manager (1) |
| | Acting Development Manager (1) |
| | Developers (5) |
| | Quality Engineer (1) |
| | Intern (1) |
| ***Development Process*** | |
| | Kanban (until recently Scrum) |
| ***Tools used*** | |
| | VersionOne |
| ***Velocity*** | |
| | The team previously had a velocity of 13 story points. |
| | Since they have moved away from a Scrum-like process, they no longer track their velocity. |
| ***Iteration length*** | |
| | Previously 3 weeks |
| | They do not follow an iterative process anymore. |

### *6.6.4.1 Team-specific Assessment*

Applying the OPS Framework and Assessment Methodology, the overall Capability and Effectiveness scores for Team D are:

| Capability: 4.214 | Effectiveness: 3.880 |
|---|---|

Using Table 6.2 to interpret these scores, both values fall in the "***Considerable***" range (3.5 – 4.4). Hence, Team D is ***considerably*** (a) capable of implementing its adopted agile method and (b) effective in implementing that method.

#### ASSESSMENT: ADEQUACY

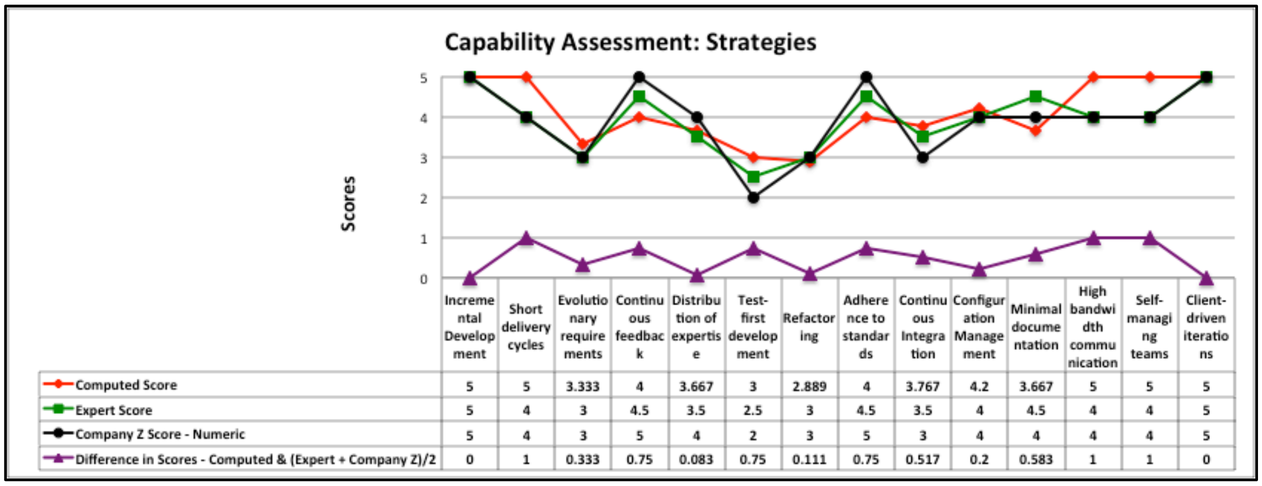
As a first step to assessing the capability and effectiveness, we assess the adequacy of the method under consideration. Team D follows Kanban. They previously used a Scrum-like process with three-week iterations. The team found Scrum to be "heavy" and intended to use a "lighter" process. Team D strives to achieve all the five objectives presented by the OPS Framework. Subsequently, from the established linkages, we know that all the nine principles identified by the Framework have to be support to achieve those objectives. By design, Kanban does not support 'Constant Development Pace.' Hence, Team D does not adhere to this principle. To achieve the other eight principles, with the exception of 'Constant Velocity', the remaining 16 strategies have to be implemented. Kanban, by design, does not tout the strategies 'Retrospection' and 'Iterative Progression.' Hence, Team D's instantiation of Kanban does not feature these strategies. Team D's adopted agile method is adequate. However, the lack of certain essential principles and strategies lowers the extent to which the corresponding objectives can be achieved. During the follow-up meetings with Team D, the team members confirmed the adequacy assessment results for their adopted agile method.

#### ASSESSMENT: CAPABILITY

Capability Assessment results for the strategies, principles, and objectives are shown in Figures 6.19, 6.20, and 6.21 respectively. From these figures, we see that the scores show similar trends.

***A Strategies Level Assessment***

Figure 6.19 shows the capability assessment results at the strategies level. Test-first Development is marginally supported. In Figure 6.19, we see lower scores for that strategy. They are currently building their environment to support the implementation of Continuous Integration. Hence, the scores reflect moderate capability with respect to Continuous Integration (see Figure 6.19). Similar to Teams B and C, in Team D, the personnel identify all the features and stories upfront. This lowers the capability to allow the requirements to evolve over time.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development | Refactoring | Adherence to standards | Continuous Integration | Configuration Management | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 5 | 5 | 3.333 | 4 | 3.667 | 3 | 2.889 | 4 | 3.767 | 4.2 | 3.667 | 5 | 5 | 5 |
| Expert Score | 5 | 4 | 3 | 4.5 | 3.5 | 2.5 | 3 | 4.5 | 3.5 | 4 | 4.5 | 4 | 4 | 5 |
| Company Z Score - Numeric | 5 | 4 | 3 | 5 | 4 | 2 | 3 | 5 | 3 | 4 | 4 | 4 | 4 | 5 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0 | 1 | 0.333 | 0.75 | 0.083 | 0.75 | 0.111 | 0.75 | 0.517 | 0.2 | 0.583 | 1 | 1 | 0 |

***Figure 6.19. Team D - Capability Assessment: Strategies***

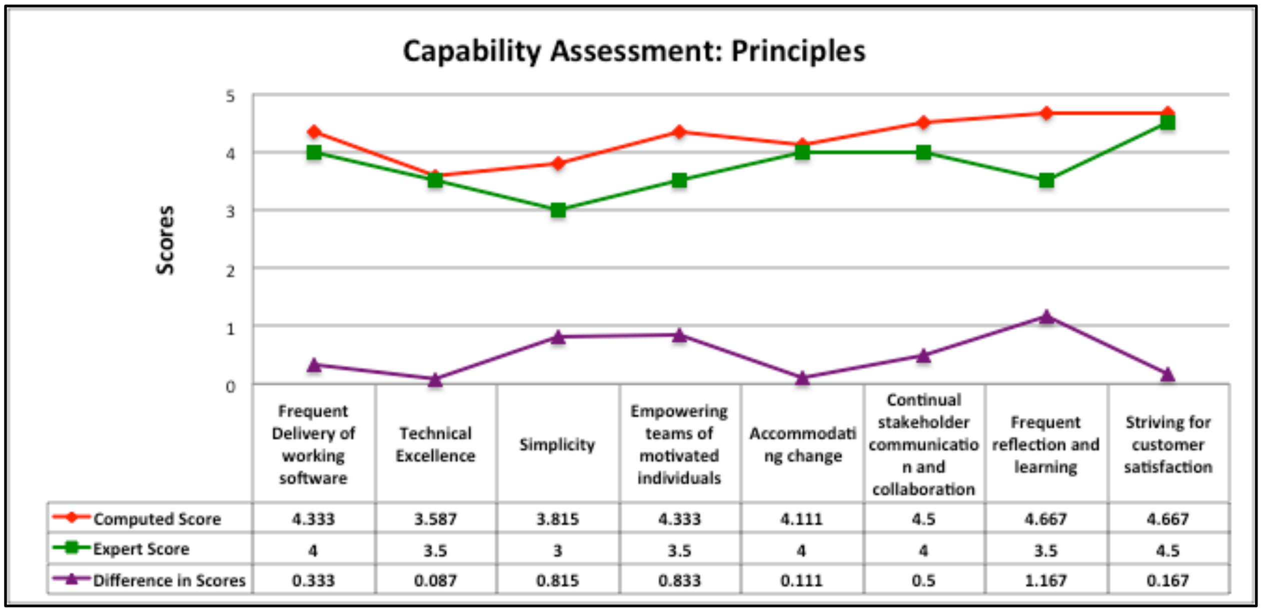
From the Pearson Correlation Coefficient values shown in Table 6.26 below, we see that the capability assessment results at the strategies level are statistically significant. We can conclude that the similarity in trends exhibited by the three sets of scores is not a chance occurrence.

***Table 6.26. Team D - Capability Assessment: Pearson Correlation Coefficient at the Strategies Level***

| Pearson Correlation At the Strategies Level – Capability Assessment |
|---|
| **Computed OPS and Expert:** |
| **r** = 0.72161, **N** = 14, **p-value** = 0.00357, two-tails |
| **Computed OPS and Company Z:** |
| **r** = 0.66158, **N** = 14, **p-value** = 0.00997, two-tails |

## *A Principles Level Assessment*

In Figure 6.20, we can observe that the Expert score for Simplicity is considerably lower. With respect to Simplicity, we have observed that Team D had worked on many features and stories in the past one year. However, these stories have not been integrated with the existing codebase and have not been deployed. That is, they focus minimally on ‘building only what is necessary’. Factoring this observation, the Expert score for Simplicity is lower. With the adoption of Kanban, the team no longer holds retrospectives. Hence, the capability to support 'Frequent Reflection and Learning' is also lowered (see Figure 6.20).



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.333 | 3.587 | 3.815 | 4.333 | 4.111 | 4.5 | 4.667 | 4.667 |
| Expert Score | 4 | 3.5 | 3 | 3.5 | 4 | 4 | 3.5 | 4.5 |
| Difference in Scores | 0.333 | 0.087 | 0.815 | 0.833 | 0.111 | 0.5 | 1.167 | 0.167 |

*Figure 6.20. Team D - Capability Assessment: Principles*

The impact of the individual OPS scores is minimized when the scores are propagated up the hierarchy. Hence, at the principles level, we are unable to show that the results are statistically significant (see Table 6.27). We also conjecture that the lack of weighted linkages in the OPS Framework also contributes to lower correlation between the two sets of scores.

*Table 6.27. Team D - Capability Assessment: Pearson Correlation Coefficient at the Principles Level*

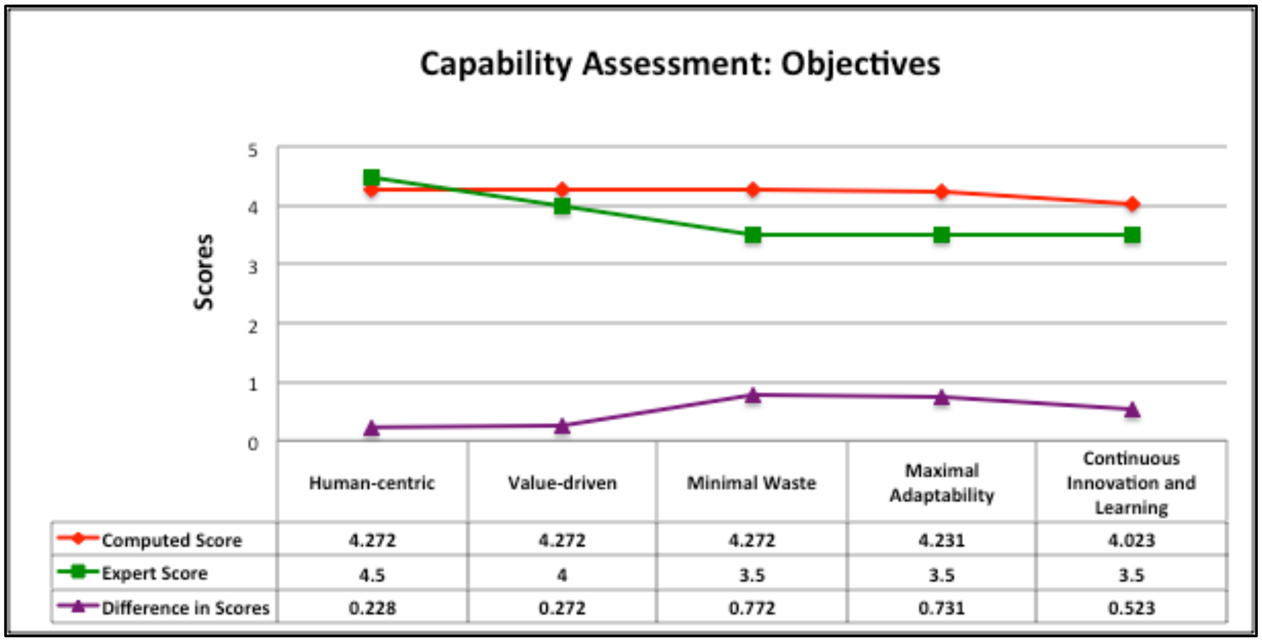
**Pearson Correlation At the Principles Level – Capability Assessment**

**Computed OPS and Expert Values**

**r** = 0.57426, **N** = 8, **p-value** = 0.13657, two-tails

***An Objective Level Assessment***

Figure 6.21 shows the capability assessment results with respect to the objectives. The Computed and Expert scores exhibit similar trends. The Expert scores are considerably lower for the objectives 'Minimal Waste' and 'Maximal Adaptability.' The team builds features that are never deployed. Also, all the features and stories are identified upfront. That is, the requirements are ‘frozen.’ These factors lower the capability of the team to maximally achieve the above-mentioned objectives.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 4.272 | 4.272 | 4.272 | 4.231 | 4.023 |
| Expert Score | 4.5 | 4 | 3.5 | 3.5 | 3.5 |
| Difference in Scores | 0.228 | 0.272 | 0.772 | 0.731 | 0.523 |

***Figure 6.21. Team D - Capability Assessment: Objectives***

With the OPS scores, we observe the ‘minimization effect’ since the impact of the individual scores is diminished as the scores are averaged at each level of the hierarchy. This minimization effect is not a factor in the Expert scores. Therefore, computing the correlation coefficient, we see that the results shown in Table 6.28 are not statistically significant.

***Table 6.28. Team D - Capability Assessment: Pearson Correlation Coefficient at the Objectives Level***

| |
|---|
| **Pearson Correlation At the Objectives Level – Capability Assessment** |
| **Computed OPS and Expert Values** |
| **r** = 0.44933, **N** = 5, **p-value** = 0.44778, two-tails |

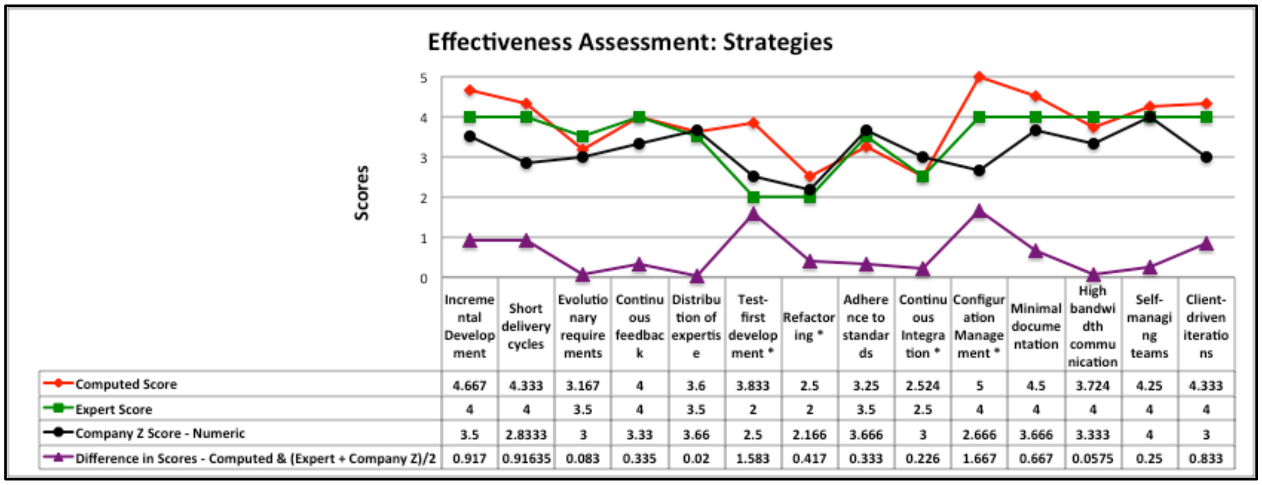
For all the four teams that we have observed, the results at the strategies level are statistically significant. At the principles and objectives levels, the scores are not statistically significant.

## ASSESSMENT: EFFECTIVENESS

Figures 6.22, 6.23, and 6.24 depict the Effectiveness Assessment results for Team D. As shown in these figures, the scores predominantly exhibit similar trends.

### *A Strategies Level Assessment*

We possess limited information to provide a complete effectiveness assessment for the strategies Test First Development, Refactoring, Continuous Integration, and Configuration Management. Hence, in Figure 6.22, we see some differences in the numeric scores, especially for the strategies Test First Development, and Configuration Management. It is to be noted that the strategies Retrospection and Iterative Progression are not included in the assessment. As discussed previously, Kanban does not advocate those strategies. Since Team D now adopts Kanban, those strategies are not implemented.



| | Incremental Development | Short delivery cycles | Evolutionary requirements | Continuous feedback | Distribution of expertise | Test-first development * | Refactoring * | Adherence to standards | Continuous Integration * | Configuration Management * | Minimal documentation | High bandwidth communication | Self-managing teams | Client-driven iterations |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.667 | 4.333 | 3.167 | 4 | 3.6 | 3.833 | 2.5 | 3.25 | 2.524 | 5 | 4.5 | 3.724 | 4.25 | 4.333 |
| Expert Score | 4 | 4 | 3.5 | 4 | 3.5 | 2 | 2 | 3.5 | 2.5 | 4 | 4 | 4 | 4 | 4 |
| Company Z Score - Numeric | 3.5 | 2.8333 | 3 | 3.33 | 3.66 | 2.5 | 2.166 | 3.666 | 3 | 2.666 | 3.666 | 3.333 | 4 | 3 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.917 | 0.91635 | 0.083 | 0.335 | 0.02 | 1.583 | 0.417 | 0.333 | 0.226 | 1.667 | 0.667 | 0.0575 | 0.25 | 0.833 |

*Figure 6.22. Team D - Effectiveness Assessment: Strategies*

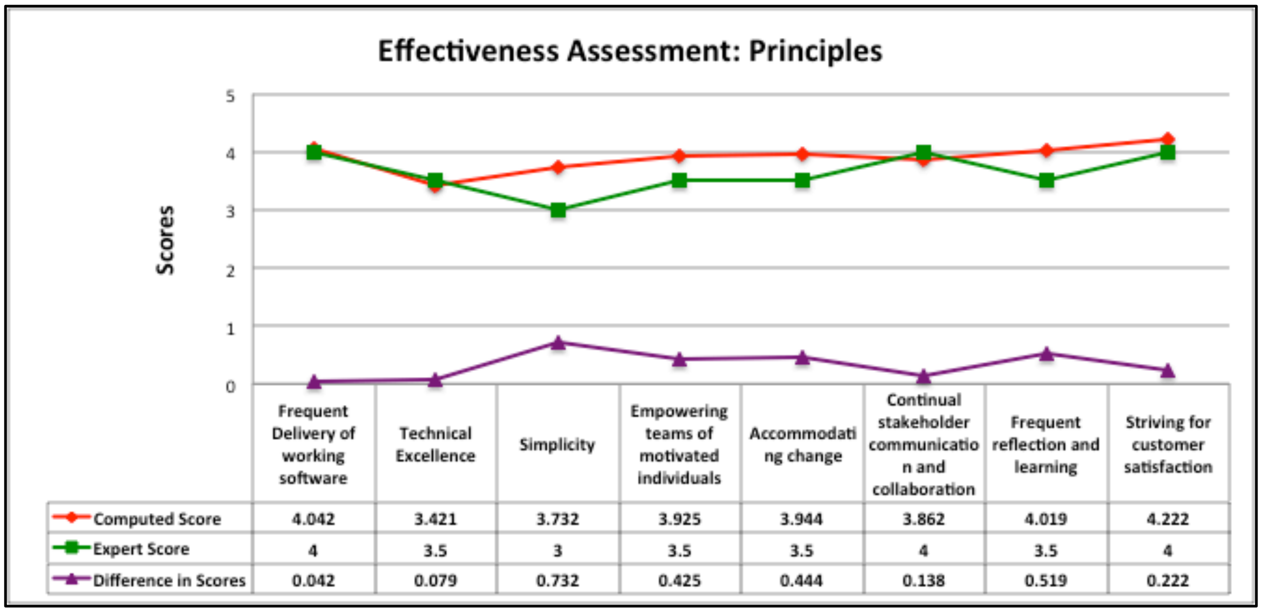
Table 6.29 below shows the r and p values for the three sets of scores at the strategies level from an effectiveness perspective. Similar to the results obtained for the other three teams, we see that the results are statistically significant with respect to the OPS and Expert scores. However, the correlation is not significant between the OPS and Company Z scores. From the survey responses recorded by Company Z personnel, we recognize that they have divergent views of the extent to which their team is effective. We surmise that we are unable to show statistical significance due to the variability in the gathered data.

***Table 6.29. Team D - Effectiveness Assessment: Pearson Correlation Coefficient at the Strategies Level***

| |
|---|
| **Pearson Correlation At the Strategies Level – Effectiveness Assessment (Preliminary)** |
| **Computed OPS and Expert:** |
| **r** = 0.71523, **N** = 14, **p-value** = 0.00403, two-tails |
| **Computed OPS and Company Z:** |
| **r** = 0.27087, **N** = 14, **p-value** = 0.34893, two-tails |

***A Principles Level Assessment***

Figure 6.23 shows the effectiveness assessment results with respect to the principles. Team D personnel often develop features that are not deployed. The code remains in the configuration management systems and is never used. Based on this observation, the Expert score for the principle ‘Simplicity’ is lower (see Figure 6.23). Also, since the team now adopts Kanban, retrospective meetings are not held. This lowers the extent to which the principle ‘Frequent Reflection and Learning’ is achieved. Hence, the Expert score for that principle is lower.



| | Frequent Delivery of working software | Technical Excellence | Simplicity | Empowering teams of motivated individuals | Accommodating change | Continual stakeholder communication and collaboration | Frequent reflection and learning | Striving for customer satisfaction |
|---|---|---|---|---|---|---|---|---|
| Computed Score | 4.042 | 3.421 | 3.732 | 3.925 | 3.944 | 3.862 | 4.019 | 4.222 |
| Expert Score | 4 | 3.5 | 3 | 3.5 | 3.5 | 4 | 3.5 | 4 |
| Difference in Scores | 0.042 | 0.079 | 0.732 | 0.425 | 0.444 | 0.138 | 0.519 | 0.222 |

***Figure 6.23. Team D - Effectiveness Assessment: Principles***

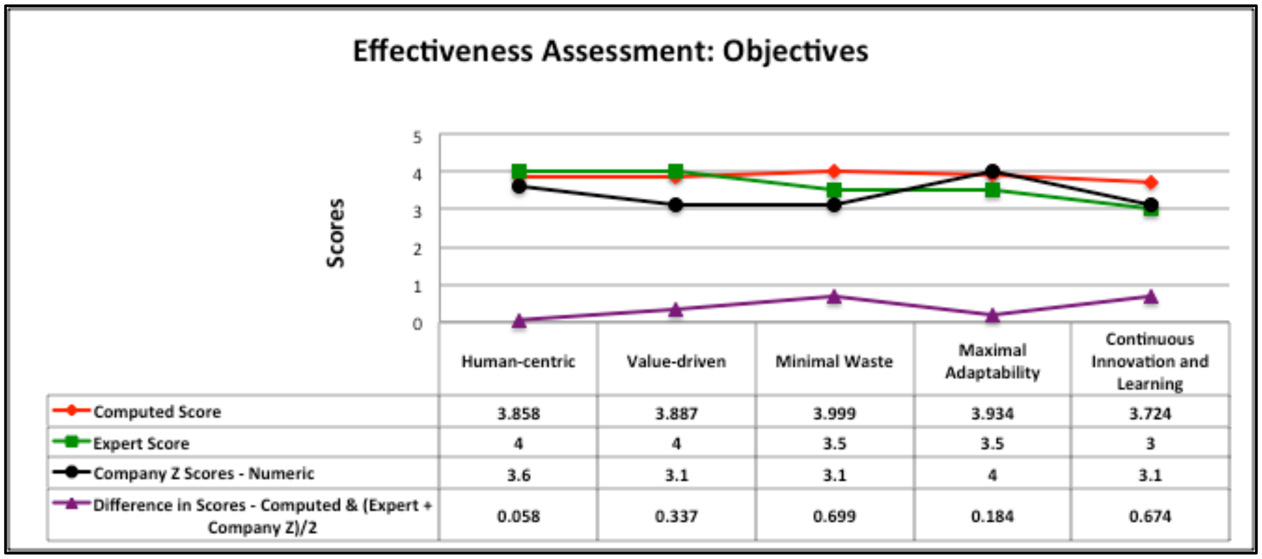
Computing the correlation between the OPS and Expert scores at the principles level from an effectiveness perspective, we see that the results are not statistically significant (see Table 6.30). As the scores are propagated up the indicator hierarchies, the impact of the individual scores is minimized. Also, the OPS scores are not weighted. We conjecture that we are unable to show statistical significance at this level due to the above-mentioned reasons. Additionally, the effectiveness assessment results are preliminary.

*Table 6.30. Team D - Effectiveness Assessment: Pearson Correlation Coefficient at the Principles Level*

| Pearson Correlation At the Principles Level – Effectiveness Assessment (Preliminary) |
|---|
| **Computed OPS and Expert Values** |
| **r** = 0.50929, **N** = 8, **p-value** = 0.19736, two-tails |

### *An Objectives Level Assessment*

In Figure 6.24, the differences in scores for the objective Value-driven stem from the differences in perceptions of the experts and the Company Z practitioners. Also, our effectiveness assessment is cursory. Team D personnel do recognize that the objective 'Minimal Waste' is achieved minimally. This is consistent with our observations and assessment with respect to that objective.



| | Human-centric | Value-driven | Minimal Waste | Maximal Adaptability | Continuous Innovation and Learning |
|---|---|---|---|---|---|
| Computed Score | 3.858 | 3.887 | 3.999 | 3.934 | 3.724 |
| Expert Score | 4 | 4 | 3.5 | 3.5 | 3 |
| Company Z Scores - Numeric | 3.6 | 3.1 | 3.1 | 4 | 3.1 |
| Difference in Scores - Computed & (Expert + Company Z)/2 | 0.058 | 0.337 | 0.699 | 0.184 | 0.674 |

*Figure 6.24. Team D - Effectiveness Assessment: Objectives*

Table 6.31 shows the computed correlation results for the three sets of scores. We see that the results are not statistically significant. We conjecture that this is due to the minimization effect and the lack of weights in the OPS Framework. Also, given the short duration of the onsite study, the effectiveness assessment results are preliminary.

***Table 6.31. Team D - Effectiveness Assessment: Pearson Correlation Coefficient at the Objectives Level***

**Pearson Correlation At the Objectives Level – Effectiveness Assessment (Preliminary)**

**Computed OPS and Expert:**

**r** = 0.41027, **N** = 5, **p-value** = 0.49268, two-tails

**Computed OPS and Company Z:**

**r** = 0.22128, **N** = 5, **p-value** = 0.72057, two-tails

### *6.6.4.2 Team-specific Observations*

This sub-section outlines our team-specific observations for Team D. These observations outline the positive aspects of its people, process, product, project, and environment. Additionally, we also provide a set of observations whose implications require further examination.

**Positive:**

- Team members, and in particular the developers, felt that Scrum was process-heavy. So, Team D has recently adopted a Kanban-like approach. True to Kanban, they have limits on the number of items that can remain at each stage in the development process. For example, they can have only 5 stories that are being developed, 2 stories that are being tested by the Quality Engineer, etc. Kanban seems to be working well for the team.
- The team is self-managing.
- The Product Owner frequently attends the daily stand-ups and is aware of the team's progress.
- Collective code ownership and knowledge sharing is promoted.
- The Product Owner maintains a continuous feedback loop with the customers.
- The Technical Product Owner maintains a Technical Debt Backlog. The development team works on the technical debt stories during the course of the release cycles.
- They focus on building the Minimum Viable Product.
- The developers estimate the effort required for each story and determine the priorities at the story level. Also, they constantly reassess the estimates and priorities of the features.
- Though the Scrum Master was relatively new to the team, she recognized the issues with their process and initiated changes for the team's betterment.

- Team D places high priority on the technical debt and support stories, which are different from the priorities set by the Product group. Efforts are underway to reconcile the two sets of priorities.
- They have made significant progress in implementing Continuous Integration.
- The Scrum Master and the Technical Product Manager ensure that they plan differently in order to avoid Quality Engineering bottlenecks towards the end of a release cycle.
- They constantly reassess the priorities of stories and other issues.

**Requires Scrutiny**

- The team has decided to not hold retrospective meetings anymore since they have moved away from Scrum. However, they do recognize the value of the retrospectives and are discussing the possibility of scheduling a retrospective meeting every month.
- The developers do not have the opportunity to interact with the customers directly. However, their Product Owner shares feedback from the customers and the users with the team.
- Test-first development is implemented only marginally.
- Team D used to follow a Scrum-like process. They had 3-week iterations. However, they have not been able to meet their commitments. That is, they would have some work that was left undone during an iteration and they would carry it over to the next. It is our observation that the team faced the issue of not meeting commitments because they did not estimate accurately.
- Occasionally, it seems that this team does not view itself as "one unit" and that members do not always work together well.
- To some extent, the developers lack direction since Team D does not have a Development Manager. They are looking to hire new personnel.
- Though story point estimates include the effort required for testing, the developers offer minimal support to the Quality Engineer.
- The developers are hesitant to help the Quality Engineer with testing when they have completed the tasks that they had committed to for that release.
- In the past, due to poor planning and estimation, the team has been unable to complete the tasks that they had committed to during an iteration. In fact, the estimates provided were "30-50% off".
- Every other team except Team D has a Subject Matter Expert from the Engineering team working closely with them.
- The team members spend considerable amounts of time fixing the production environment issues.

- Due to inaccurate estimates provided by the development team, it was discussed that they may have to schedule another release planning meeting.
- Team D does not include a User Experience designer. They "plug into" Microsoft's usability or rely on the expertise of User Experience experts from other teams.

#### *6.6.4.3 Summary*

Team D's instantiation of Kanban is adequate. By design, Kanban does not advocate the principle 'Constant Development Pace' and the strategies 'Constant Velocity', 'Retrospection' and 'Iterative Progression.' Hence, the above-mentioned elements are not included in the assessment of adequacy, capability, and effectiveness. The lack of these elements lowers the extent to which certain objectives are achieved. Team D personnel concur with our assessment of the adequacy of their adopted agile method.

Similar to the other three teams, the overall OPS scores for Team D from both the capability and effectiveness perspectives fall in the 'considerable' range. The capability results are statistically significant at the strategies level. From an effectiveness perspective, the correlation between the OPS and Expert scores is statistically significant at the strategies level.

***The team members confirmed our findings during the follow-up meeting. They agreed with the OPS scores and with the team-specific observations.*** They recognize the importance of 'Retrospection' and are looking to incorporate that strategy in their process. The development team is happy to have moved away from Scrum. However, it remains to be seen if a Kanban-like approach helps the team achieve its goals and increase its productivity.

## 6.6.5 OPS Related Observations common to all the four teams

During the course of our on-site study, we gained insights about the development processes being used, the development environments, the people, the projects, the company culture, etc. We attended several meetings, interacted with practitioners, and observed teams in action. In this section, we outline the aspects that are in common to all the four teams (Teams A, B, C, and D) under consideration. These observations showcase the positive aspects and those that require scrutiny.

**Positive:**

- People are valued more than the processes and tools.
- They work towards maximizing value to all the stakeholders involved.
- The teams are adaptable with respect to accommodating changes to requirements, fine-tuning their process, work estimates, schedules, etc.
- The organization provides an open physical environment.
- The teams are encouraged to maintain minimal documentation.
- Refactoring is strongly encouraged. However, they find it difficult to refactor legacy code.
- Each code fragment is peer-reviewed by one or more developers. This is carried out in order to ensure that the code meets accepted standards and to create shared knowledge.
- Open communication is encouraged.
- All the teams hold a stand-up meeting everyday.
- The developers and the QE personnel estimate the story points for the stories.
- Cross team collaboration is encouraged.
- In Teams A, C, and B, code is extensively tested. Hence, the number of bugs reported post-deployment is significantly minimized.
- The developers work in pairs on an as needed basis.
- In Teams A, C, and B, developers and the quality engineers contribute equally to testing the systems.
- The teams manage risk by deploying software in stages.

**Requires Scrutiny:**

- Currently, the teams do not track the number of bugs discovered during pre- and post- deployment. That is, they do not examine the effectiveness of their testing process. The teams recognize that issue and are working to address it.
- The Capability for automated testing is minimal. Plans are being made to move to toward a more automated testing environment. As a first step, they are hiring personnel to develop automated tests and testing frameworks.
- Though the developers recognize the value of writing tests before writing code, the teams have not been successful in adopting TDD.
- The organization is looking to move towards Continuous Integration and Continuous Delivery (CICD). The teams are working to set up their environments to support those strategies. However, Teams AppsCP, Billing, and MSFT Services share their code base. The shared code base could potentially preclude the adoption of CICD.
- Inter-team communication is sometimes ineffective especially in the case of collaborating with geographically distributed teams.
- More often than not, meetings are rescheduled. The team members get frustrated due to constant changes to their schedule.

### 6.6.6 Comments on value/score nuances

Predominantly, all three sets of scores (computed, expert, and Company Z) show ***similar trends***. Though the numeric values are not identical, the differences in the values are more often than not, negligible.

We conjecture that the differences in the three sets of scores stem from the following:

1. Our perceptions of the objectives, principles, and strategies may differ from those of the practitioners.
2. We possess sufficient data to provide only a preliminary assessment of the effectiveness. As discussed previously, assessing effectiveness is a longitudinal process. Our assessments are based on what we have observed during the three months when we were on-site. Company Z personnel have based their evaluations on their insights gained over a considerable period of time.
3. The questions associated with the indicators are not an exhaustive set. Therefore, the computed and the expert scores may differ in numerical value.

4. The members of the development teams have had limited contact with the external customers and users. Hence, for those strategies that address communication and collaboration with the customers, their scores are inconsistent with the other sets of scores.

5. The linkages in the OPS Framework are weighted equally. The experts and the practitioners from Company Z may have valued certain aspects of the people, process, projects, products or the environment more than the others. These weights may have contributed to a lower or greater numeric value than the computed OPS based scores.

## *6.7 Summary and Confirmation of Findings*

To validate the results obtained by applying the OPS Framework and assessment methodology in Company Z, we envisioned the need to gather feedback from that organization's personnel. Hence, as a final step to the study at Company Z, we compiled a report describing the assessment process, team-specific results and observations, and elements specific to Company Z. In addition to the report, we held follow-up meetings with the individual teams and with management personnel to confirm our findings and discuss the implications of our observations.

### 6.7.1 Confirmation by Individual Teams

Each of teams under consideration – Teams A, B, C, and D were provided with a ***team-specific report*** describing the

- OPS Framework and assessment methodology
- Study objectives and mechanics
- Scoring scales
- Team-specific results – OPS-based, Expert, and scores assigned by personnel of that team
- Team-specific observations

It is to be noted that each team had access ***only*** to the results and observations specific to their team.

Prior to assessing capability and effectiveness, we also examined the adequacy of each of the methods under consideration. During the follow-up meetings, the team personnel confirmed our perceptions of the composition of their adopted agile methods and the adequacy assessment results.

The capability and effectiveness results include three sets of scores. In addition to the OPS based scoring, we include scores assigned by the SMEs and Company Z personnel. These additional scores have been recorded for both capability and effectiveness. The SME scores are independent of the OPS and Company Z scores. These scores were assigned ***before*** computing the OPS scores and compiling the Company Z scores. In general, all three sets of scores follow ***similar trends***. Differences between these three sets of scores are negligible, and in particular, at the strategies level. More specifically, at the strategies level, the results are ***statistically significant***. That is, the similarity in trends exhibited by the three sets of values is not by random chance.

***The team members of each of the four teams confirmed the OPS-based scores and team-specific observations.*** In the follow-up meetings, the team-specific observations were addressed and solution approaches were discussed. The teams recognize that certain aspects of their process need to be reviewed for their impact and are currently involved in addressing those issues.

## 6.7.2 Confirmation of Findings by Higher level Management

Prior to holding follow-up meetings with the individual team, we met with management personnel to present the comprehensive report and gather feedback about our findings. The report included the OPS-based analysis and findings for all the four teams. The intent was to gather feedback about the OPS-based results from an organizational perspective.

From the computations, the overall Capability and Effectiveness scores (OPS-based) recorded on a five-point scale for the four teams are given in Table 6.8.

***Table 6.8. Overall OPS-based Capability and Effectiveness scores for the four teams***

| Teams | Capability | Effectiveness |
|---|---|---|
| A | 4.395 | 4.133 |
| B | 4.257 | 3.941 |
| C | 4.227 | 3.886 |
| D | 4.214 | 3.880 |

Based on the scale given in Table 6.2, the Capability and Effectiveness scores (see Table 6.8) for all the four teams fall in the “Considerable” range (3.5 – 4.4). That is, we can conclude that Teams A, B, C, and D are ***considerably*** (a) capable of implementing their adopted agile method and (b) effective in implementing that method.

During the course of our on-site study, we gained insights about the development processes being used, the development environments, the people, the projects, the company culture, etc. We attended several meetings, interacted with practitioners, and observed teams in action. In addition to the OPS-based results, the report also included certain aspects that are in common to all the four teams (Teams A, B, C, and D) under consideration. These observations showcase the positive aspects and those that require scrutiny.

Additionally, the data collected also includes elements specific to Company Z. The implications of those elements listed below require further scrutiny:

- Velocity and estimation schemes used by the teams
- Appropriateness of the composition of teams
- Interactions between the development teams and the external customers
- Support for analytics (Example: tracking the number of bugs uncovered pre- and post- deployment)
- Support for CICD (in particular, the conduciveness of the development environment for supporting CICD)
- Scheduling and duration of meetings (frequent rescheduling and meetings run over time)
- Inter- and intra-team communication (especially with geographically distributed teams)
- Striving to maximize value to the customers
- Maintaining a constant development pace and 40 hour work weeks
- Support for User Experience research, design, and testing

***During the follow-up meeting with the management personnel, they agreed with the OPS scores for capability and effectiveness for all the four teams. They confirmed that the capability and effectiveness of those teams are 'considerable.'*** Additionally, the implications of the team-specific observations and the above-mentioned elements specific to Company Z were discussed. ***The management personnel concurred that those observations are possible inadequacies that have to be addressed in order for the teams to be maximally capable and effective.***

# 7 Conclusions

The creation of the OPS Framework and assessment methodology presented in this dissertation has been motivated by the need for a structured, systematic, and comprehensive approach to assessing the 'goodness' of agile methods. Existing agile assessment approaches are limited in scope and application. That is, most assessment approaches focus on the product and assess in part, certain aspects of the development process. Our inclusive approach focuses on the people, process, product, project, and environment characteristics.

We assess 'goodness' by examining (a) the adequacy of a method, (b) the capability of an organization to provide the supporting environment for implementing the method, and (c) the effectiveness of that method. To guide our assessment, we employ the Objectives, Principles, Strategies (OPS) Framework. The Framework identifies objectives, principles, strategies, and indicators. Definitive linkages between the components have also been established to support the assessment process.

Currently, we have substantiated the components of the OPS Framework and our approach to assessing the adequacy, capability, and effectiveness of agile methods. To establish the viability and validity of the OPS Assessment approach, we implemented a study at Company Z, Blacksburg. During the three-month onsite study, we examined the agile methods adopted by four different teams in Company Z, to assess the 'goodness' of those agile methods primarily from the adequacy and capability perspectives. Our assessment of effectiveness is preliminary. The assessment results and findings are consistent with the perceived reality.

Our inclusive *assessment approach can be used by organizations to assess their adopted agile method, identify the effective components of their development process, as well as reveal possible inadequacies, and potential enhancements.* In this chapter, we discuss the main contributions of the research presented in this dissertation and future directions.

## 7.1 Main Contributions

The OPS Framework and assessment approach contribute to the Software Engineering body of knowledge and in particular to the field of Agile Assessment and Software Process Improvement. With the number of organizations adopting agile methods increasing, there is a need to determine if those adopted agile methods help achieve organizational objectives and yield expected results. The work presented in this dissertation provides the agile community with a structured, systematic, and comprehensive approach to assessing agile methods. The contributions discussed in this section stem from the OPS Framework and assessment methodology.

1. **Guiding Framework for Assessing Agile Methods**

   Agile methods provide practitioners the flexibility to adopt principles and strategies best suited to their needs. While that flexibility is consistent with the agile philosophy, more often than not, practitioners implement strategies and adhere to principles that may be sub-optimal relative to achieving desired objectives. The OPS Framework presented in this research has been created to reflect the viewpoint that any viable agile method (a) strives to achieve a set of *objectives*, (b) adheres to a set of *principles* that govern the achievement of those objectives, and (c) implements *strategies* that are reflective of those principles. Also, linkages between the objectives and principles, and principles and strategies have also been established to depict the relationships among them. The OPS Framework presents a structured and hierarchical view of the principles and strategies that are essential to achieve the set of desirable objectives.

   In accordance with the above, the OPS Framework identifies the following components:

   (a) ***Objectives***

   At the first level, the Framework identifies five objectives that reflect the agile philosophy. These objectives have been derived from the manifesto values and have been substantiated by practitioners of the agile community. This set of acceptable objectives, captures the current state of the agile philosophy, its values, and the development paradigm.

   (b) ***Principles***

   Principles govern the achievement of the objectives. Defined at the second level in the hierarchy, the nine identified principles derived from the agile manifesto, present a comprehensive and consistent view of agile software development. Agile practitioners have confirmed that an agile software development process should maximally promote the nine principles identified by the OPS Framework.

(c) ***Strategies***

At the third level, the Framework presents strategies that are implementations of the principles. These strategies, defined at a lower level of abstraction than the objectives and principles, are tangible concepts that can be adopted by practitioners. The 17 strategies currently included in the OPS Framework is not an exhaustive set.

(d) ***Linkages***

As mentioned previously, to achieve an objective, a certain set of principles has to be adhered to. Similarly, to implement a principle, particular strategies have to be adopted. The definitive linkages established by the OPS Framework represent these relationships between the objectives, principles, and strategies. These linkages have been substantiated through an extensive literature review, discussions with practitioners, and our observations.

The OPS Framework guides the assessment process described in this dissertation. Additionally, it contributes to the agile community by providing a structured view of the agile software development paradigm.

**2. Unifying View of the Agile Software Development Paradigm**

In today's agile software development environment, the values and principles stated in the agile manifesto are often misunderstood or misconstrued by organizations and practitioners. To a large extent, this can be attributed to the current state of the agile software development, which demands a more comprehensive and detailed understanding of what it really means to be agile. The Agile Manifesto offers a holistic view of the agile philosophy. It serves as a starting point for understanding the notion of agility. Nevertheless, since the creation of the manifesto, the values and principles that underlie the agile methodologies have undergone constant evolution and refinement. Therefore, we need an alternate *unifying* view of agile software development that reflects the current state of agile methodologies and practices.

The OPS Framework described in this dissertation provides that unifying view of agile software development. The Framework is designed to reveal the common themes that underlie the existing agile methodologies by identifying *objectives* that are reflective of the agile goals and philosophy, *principles* that govern the achievement of those objectives, and *strategies* that are implementations of those principles. Because there is a wide range of agile methods from which to choose, practitioners are often challenged when identifying the appropriate set of principles and strategies they need in order to achieve desired objectives. To address this issue, the OPS Framework also provides relationships or *linkages* among its identified objectives, principles and practices. The objectives, principles and practices, together with the linkages among them, provide a holistic view of the agile philosophy.

**3. Assessment Process examining the 'Goodness' of Agile Methods**

The agile philosophy places utmost importance on "working software." Hence, most agile methods focus on building a quality product, on time, and within budget. Consequently, existing agile assessment approaches present assessment criteria that primarily focus on the product, thereby ignoring measures examining the people, process, or project. Those few approaches that evaluate the process are limited in scope and application. More specifically, existing tools and processes focus on specific aspects of the development process. Therefore, there is a need for a comprehensive approach to assessing agile methods.

In this research, we present a systematic and comprehensive approach to assessing agile methods. To facilitate the assessment process, we present three complementary views – Adequacy, Capability, and Effectiveness that are defined as follows:

Adequacy: The sufficiency of an agile method with respect to achieving its stated objectives.

Capability: The ability of an organization or a team to support the implementation of their adopted agile method(s).

Effectiveness: The extent to which the agile method under consideration produces the expected results.

The above-mentioned three perspectives present a comprehensive approach to assessing agile methods. The OPS Framework guides the assessment process. We assess adequacy by employing a top-down traversal of the linkages from the objectives to the principles, and from the principles to the strategies. To assess capability and effectiveness, we follow the linkages in a bottom-up fashion from the strategies to the objectives. This bottom-up traversal follows a top-down traversal examining the adequacy. More specifically, prior to assessing capability and effectiveness, we determine the adequacy of the method under consideration. These traversals – both top-down and bottom-up, present a systematic approach to assessing agile methods.

In addition to providing a systematic and comprehensive approach, the OPS-based assessment methodology is ***viable*** and ***valid***. We have implemented the Framework and assessment methodology in an industry setting at Company Z. The onsite study was minimally intrusive and was completed in a short timespan of three months. This shows that our assessment process is a viable approach to assessing agile methods. Also, the OPS-based results from the study were consistent with the practitioners' (Company Z personnel) views of their adopted agile methods. More specifically, members of the four teams and the management personnel confirmed the OPS-based results and findings. Hence, the OPS Framework and assessment methodology is a ***valid*** approach to assessing the 'goodness' of agile methods.

A significant contribution to the agile community is an agile assessment approach that is systematic, comprehensive, viable, and valid.

**4. Assessment Criteria for Assessing Agile Methods**

To assess agile methods, the mechanics of implementation of those methods and the environments within which those methods are applied have to be studied. The objectives, principles, and strategies presented are all described at higher levels of abstraction. Hence, to assess agile methods, we should examine characteristics that are observable. When used, the strategies induce sets of observable characteristics. These characteristics are properties of the people, process, project, product, and the environment. Hence, to facilitate our assessment process, we identify observable properties of the people, process, project, product, and the environment.

The assessment of capability and effectiveness is predicated by the identification of these properties. For capability, we examine the people, process, project, and the environment properties. Product properties and process artifacts are included in the effectiveness assessment process.

Each strategy has an associated set of properties. Each (strategy, property) pair is an indicator. These indicators represent measurable elements of an agile method. Associated with each indicator is a set of questions that help measure that indicator. In this dissertation, we present indicators that are specific to assessing capability and effectiveness. Currently, the OPS Framework identifies 80 indicators and an associated set of 150 questions that measure those indicators. We have constructed indicator hierarchies for both capability and effectiveness. Using these indicators, practitioners can examine the capability of their team or organization and the effectiveness of their adopted agile methods. These indicators contribute to the body of knowledge pertaining to post-adoption agile assessment.

## *7.2 Future Directions*

The OPS Framework and assessment methodology presented in this dissertation are a first step towards providing a systematic, structured, and comprehensive approach to assessing the 'goodness' of agile methods. We recognize that the Framework and the methodology must evolve based on future research findings. The following are some future directions for the work presented in this dissertation:

1. **Assigning weights to the linkages**

   We realize that for an objective and the set of principles associated with that objective, one or more principles may support that objective to a greater extent than the others. Similarly, for each principle and the set of strategies that are related to that principle, some strategies may be necessary for the achievement of that principle. Recognizing that some relationships between the objectives, principles, and strategies may be more important than the others, we have to address this disparity in our solution approach.

   Recall that the OPS Framework in its current state assigns equal weights to all the linkages. The properties associated with the strategies are also weighted equally. We envision refining the Framework to include weighted linkages.

2. **Establishing statistical significance at the objectives and principles level**

   We have designed and implemented a study at Company Z to validate the assessment approach. We have assessed the adequacy, capability, and effectiveness (preliminary), of agile methods adopted by four different teams in that organization. The results include three sets of scores – Computed OPS-based scores, scores assigned by SMEs and by Company Z personnel (see Chapter 7). All three sets of scores mostly show similar trends. From the data analysis results, we see that the results are statistically significant at the strategies level. That is, the results show that the correlation between the three sets is not a chance occurrence. However, we are unable to prove statistical significance at the objectives and principles levels.

   As the scores are propagated up the OPS Framework, the impact of the individual scores is minimized. We conjecture that we are unable to prove statistical significance at the objectives and principles levels due to this “minimization effect.” Additionally, we surmise that the lack of weighted linkages in the OPS Framework could be a contributing factor to the “minimization effect.” One of the goals for the future is to investigate the reason for this effect and confirm our hypothesis.

3. **Implementing a longitudinal study for assessing effectiveness**

   As discussed previously, evaluating effectiveness is a longitudinal process. Given the short duration of the onsite study at Company Z, our effectiveness assessment results are preliminary. Hence, a more long-term goal is also to validate the OPS Framework and its assessment process as it relates to evaluating the effectiveness of an instantiated agile method. This would, of course, require a longitudinal study, but is also necessary to fully validate the assessment process.

4. **Automating the assessment process**

   Future work may include developing automated tools for OPS-based agile assessment. We envision developing a web-based system that would (a) facilitate data collection and (b) assess agile methods based on the input provided.

5. **Revisiting the Objectives, Principles, Strategies, and Linkages**

   The objectives, principles, and strategies identified by the OPS Framework are reflective of the current state of agile software development. We recognize that these components would evolve in the future. Hence, it is necessary to periodically revisit the identified objectives, principles, and strategies to ensure that they are relevant. Similarly, we also realize that the necessity and sufficiency of the set of established linkages have to be reexamined regularly.

6. **Expanding the indicator set**

   The identified indicators reflect aspects of the people, process, project, product, and the environment. However, these indicators reflect a subset of the collection of all possible indicators. Another task for the future is to reexamine the indicator hierarchies and potentially expand them to provide a comprehensive set of measurement indicators.

## *7.3 Summary*

Our research is motivated by the need for a comprehensive approach to assessing the ‘goodness’ of agile methods. We assess ‘goodness’ by examining (a) the adequacy of a method, (b) the capability of an organization to provide the supporting environment for implementing the method, and (c) the effectiveness of that method. To guide our assessment, we present the OPS Framework. The Framework identifies objectives, principles, strategies, and properties. Definitive linkages between the components have also been established to support the assessment process. Currently, we have substantiated the components of the OPS Framework and our approach to assessing the ‘goodness’ of agile methods. We have implemented the OPS Framework and assessment methodology in Company Z, Blacksburg. The OPS-based results and findings are consistent with the practitioners’ perceptions.

While, we recognize that the OPS Framework and assessment methodology must evolve based on future research findings, we are encouraged by the feedback provided by members of the agile community. The practical implementation of our work in an industry setting shows that the Framework and methodology are viable. Confirmation of the OPS-based results and findings attest to the validity of the OPS Framework and assessment methodology.

# Bibliography

P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta (2002), *Agile Software Development Methods: Review and Analysis*. Finland: VTT Publications.

D. M. Ahern, A. Clouse, and R. Turner (2008), *CMMI Distilled [Electronic Resource]: A Practical Introduction to Integrated Process Improvement*, 3rd ed. Upper Saddle River, N.J.: Addison-Wesley.

S. W. Ambler (2002), *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. New York: John Wiley & Sons, Inc.

S. W. Ambler (2013a), *Requirements Envisioning: An Agile Best Practice*, http://www.agilemodeling.com/essays/initialRequirementsModeling.htm

S. W. Ambler (2013b), *Agile Requirements Modeling*, http://www.agilemodeling.com/essays/agileRequirements.htm

S. W. Ambler (2013c), *The "Broken Iron Triangle" Software Development Anti-Pattern*, http://www.ambysoft.com/essays/brokenTriangle.html

S. W. Ambler (2013d), *Agile/Lean Documentation: Strategies for Agile Software Development*, http://www.agilemodeling.com/essays/agileDocumentation.htm

S. W. Ambler (2013e), *Agile Requirements Change Management*, http://www.agilemodeling.com/essays/changeManagement.htm

J. D. Arthur and R. E. Nance (1991), "A Framework for Assessing the Adequacy and Effectiveness of Software Development Methodologies," Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

J. D. Arthur, R. E. Nance, and S. M. Henry (1986), "A Procedural Approach to Evaluating Software Development Methodologies: The Foundation," Technical Report, Department of Computer Science, Virginia Tech, Blacksburg, VA.

G. Asproni (2004), Motivation, Teamwork, and Agile-Development. *Agile Times*.

O. Balci (2008), "Collaborative Evaluation Environment: Methodology and Web-Based Software," CS 6704 Class Notes, Fall 2008, Department of Computer Science, Virginia Tech, Blacksburg, VA.

O. Balci, R. J. Adams, D. S. Myers, and R. E. Nance (2002), "Credibility Assessment: A Collaborative Evaluation Environment for Credibility Assessment of Modeling and Simulation Applications," *in Proceedings of the 34th conference on Winter Simulation: Exploring New Frontiers*, San Diego, California, pp. 214-220.

O. Balci and W. F. Ormsby (2008), "Network-Centric Military System Architecture Assessment Methodology," *International Journal of System of Systems Engineering,* vol. 1, pp. 271-292.

V. R. Basili, G. Caldiera, and H. D. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*: John Wiley & Sons, Inc., 1994.

K. Beck (2003), *Test-Driven Development: By Example*: Addison-Wesley.

K. Beck and C. Andres (2004), *Extreme Programming Explained: Embrace Change*, 2 ed.: Addison-Wesley Professional.

K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn*, et al.* (2001), *Manifesto for Agile Software Development*, www.agilemanifesto.org

J. Bergin, J. Caristi, Y. Dubinsky, O. Hazzan*, et al.* (2004), "Teaching Software Development Methods: The Case of Extreme Programming," in *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, Norfolk, Virginia, USA, pp. 448-449.

J. Bergin, C. Kussmaul, T. Reichlmayr, J. Caristi*, et al.* (2005), "Agile Development in Computer Science Education: Practices and Prognosis," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, St. Louis, Missouri, USA, pp. 130-131.

B. Boehm (1988), "A Spiral Model of Software Development and Enhancement," *Computer,* vol. 21, pp. 61-72.

B. Boehm (1991), Software Risk Management: Principles and Practices. *IEEE Software*. 32-41

B. Boehm, A. Egyed, J. Kwan, D. Port*, et al.* (1998), "Using the Winwin Spiral Model: A Case Study," *Computer,* vol. 31, pp. 33-44.

B. Boehm and R. Turner (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*: Addison-Wesley.

E. J. Braude (2001), *Software Engineering: An Object-Oriented Perspective*: John Wiley & Sons, Inc.

F. P. Brooks, Jr. (1987), "No Silver Bullet Essence and Accidents of Software Engineering," *Computer,* vol. 20, pp. 10-19.

D. Bustard and F. Keenan (2009), "Soft Systems Methodology: An Aid to Agile Development?," in *Information Systems Development*, W. Wojtkowski, G. Wojtkowski, M. Lang, K. Conboy, and C. Barry, Eds.: Springer US, pp. 25-38.

L. Cao and B. Ramesh (2008), Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*.

J. P. Cavano and J. A. McCall (1978), "A Framework for the Measurement of Software Quality," presented at the Proceedings of the software quality assurance workshop on Functional and performance issues.

A. Chigani (2010), "Agile Architecting: Using Agile Principles to Agilitize the Architecting Process," in *6th Software Architecture Technology User Network Conference (SATURN 2010)*, Minneapolis, MN.

A. Cockburn (2002), *Agile Software Development*: Addison-Wesley.

A. Cockburn (2004), *Crystal Clear: A Human-Powered Methodology for Small Teams*: Addison-Wesley Professional.

A. Cockburn and J. Highsmith (2001), "Agile Software Development, the People Factor," *Computer,* vol. 34, pp. 131-133.

M. Cohn (2004), *User Stories Applied: For Agile Software Development*: Addison Wesley Longman Publishing Co., Inc.

M. Cohn (2006), *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

S. D. Conte, H. E. Dunsmore, and V. Y. Shen (1986), *Software Engineering Metrics and Models*. Benjamin/Cmmings Publishing Company, Inc.

W. Cunningham (2007), *Fit: Framework for Integrated Test*, http://fit.c2.com/

A. V. Dandekar (1987), "A Procedural Approach to the Evaluation of Software Development Methodologies," Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA.

A. M. Davis (2003), "The Art of Requirements Triage," *Computer,* vol. 36, pp. 42-49.

E. Deming (1986), *Out of the Crisis*. Cambridge, MA: MIT Center for Advanced Engineering Study.

M. Denne and J. Cleland-Huang (2004), *Software by Numbers: Low-Risk, High-Return Development,* Upper Saddle River, NJ: Prentice Hall PTR.

J. Dietrich and A. Paschke (2005), "On the Test-Driven Development and Validation of Business Rules," in *4th International Conference on Information Systems Technology and its Applications (ISTA'2005)*, Palmerston North, New Zealand, pp. 31-48.

T. Dingsøyr and G. K. Hanssen (2003), "Extending Agile Methods: Postmortem Reviews as Extended Feedback," in *Advances in Learning Software Organizations*. vol. 2640, S. Henninger and F. Maurer, Eds.: Springer Berlin / Heidelberg, pp. 4-12.

G. Dinwiddie (2009), *Diy Project/Process Evaluation Kit*, http://blog.gdinwiddie.com/2009/08/18/diy-projectprocess-evaluation-kit/

M. Drury, K. Conboy, and K. Power (2011), "Decision Making in Agile Development: A Focus Group Study of Decisions and Obstacles," in *Agile Conference (AGILE) 2011*, pp. 39-47.

N. E. Fenton and M. Neil (2000), "Software Metrics: Roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland.

P. C. Fishburn (1967), "Additive Utilities with Incomplete Product Sets: Application to Priorities and Assignments," *Operations Research,* vol. 15, pp. 537-542.

E. H. Forman and S. I. Gass (2001), "The Analytic Hierarchy Process - An Exposition," *Operations Research,* vol. 49,

M. Fowler (2006a), *Continuous Integration*, http://www.martinfowler.com/articles/continuousIntegration.html

M. Fowler (2006b), *Semantic Diffusion*, http://martinfowler.com/bliki/SemanticDiffusion.html

A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti*, et al.* (1998), "Applying Gqm in an Industrial Software Factory," *ACM Transactions on Software Engineering and Methodology,* vol. Vol. 7,

H. Glazer, J. Dalton, D. Anderson, M. Konrad, *et al.* (2008), "CMMI or Agile: Why Not Embrace Both!," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note CMU/SEI-2008-TN-003, http://www.sei.cmu.edu/library/abstracts/reports/08tn003.cfm

M. K. Groener (2002), "Capturing Requirements Meeting Customer Intent: A Methodological Approach," Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA.

M. K. Groener and J. D. Arthur (2004), "An Operational Model for Structuring the Requirements Generation Process " *Requirements Engineering,* vol. 10.

B. Hanks, C. Wellington, T. Reichlmayr, and C. Coupal (2008), "Integrating Agility in the CS Curriculum: Practices through Values," in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, Portland, OR, USA, pp. 19-20.

A. M. J. Hass (2003), *Configuration Management Principles and Practice*. Boston, MA: Addison-Wesley.

J. Hedman and M. Lind (2009), "Is There Only One Systems Development Life Cycle?," in *Information Systems Development*, W. Wojtkowski, G. Wojtkowski, M. Lang, K. Conboy, and C. Barry, Eds., ed: Springer US, pp. 105-116.

P. Henderson (2006), “Why Large It Projects Fail,” http://users.ecs.soton.ac.uk/ph/LargeIT.pdf

J. Highsmith (2002), *Agile Software Development Ecosystems*. Addison-Wesley.

J. Hunt (2006), *Agile Software Construction*. London: Springer-Verlag.

*EEE Standards Collection: Software Engineering* (1993), IEEE Standard 610.12-1990.

InfoQ (2007), *Interview: Agile Thought-Leader Alistair Cockburn*, http://www.infoq.com/interviews/alistair-cockburn-interview-2006;jsessionid=2C35A4BA4AF85EBFDA8B86F126A7DC81

M. James (2007), *A Scrummaster's Checklist*, http://blogs.danube.com/a-scrummasters-checklist?q=blog/michaeljames/a_scrummasters_checklist

A. Janus, A. Schmietendorf, R. Dumke, and J. Jager (2012), “The 3c Approach for Agile Quality Assurance,” in *Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on*, pp. 9-13.

R. Jeffries (2002), *What Is Extreme Programming*, http://xprogramming.com/what-is-extreme-programming

J. Johnson (2002), “The Cost of Big Requirements up Front (Bruf),” presented at the XP (eXtreme Programming) 2002 Conference, Alghero, Sardinia.

A. Koch (2004), *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House Publishers.

S. K. Land (2005), *Jumpstart CMM/CMMI Software Process Improvement: Using Ieee Software Engineering Standards*. Hoboken, NJ: Wiley.

C. Larman (2004), *Agile and Iterative Development: A Manager's Guide*. Boston: Addison-Wesley.

A. Law and R. Charron (2005), “Effects of Agile Practices on Social Factors,” in *Proceedings of the 2005 workshop on Human and social factors of software engineering*, St. Louis, Missouri, pp. 1-5.

J. C. Lee, D. Scott McCrickard, and K. T. Stevens (2009), “Examining the Foundations of Agile Usability with Extreme Scenario-Based Design,” in *Agile Conference (AGILE) 2009*, pp. 3-10.

D. Leffingwell (2007), *Scaling Software Agility: Best Practices for Large Enterprises*, 1 ed.: Addison-Wesley Professional.

R. Likert (1932), “A Technique for the Measurement of Attitudes,” *Archives of Psychology,* vol. 22, pp. 1-55.

J. Little (2007), *The Nokia Test,* http://agileconsortium.blogspot.com/2007/12/nokia-test.html

H. V. Loon (2004), *Process Assessment and ISO/IEC 15504: A Reference Book*. Springer Science.

R. C. Martin (2003), *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR.

J. McAvoy and T. Butler (2009), “A Failure to Learn in a Software Development Team: The Unsuccessful Introduction of an Agile Method,” in *Information Systems Development*, W. Wojtkowski, G. Wojtkowski, M. Lang, K. Conboy, and C. Barry, Eds., ed: Springer US, pp. 1-13.

R. McCauley (2001), “Agile Development Methods Poised to Upset Status Quo,” *SIGCSE Bulletin,* vol. 33, pp. 14-15.

R. E. Nance and J. D. Arthur (2002), *Managing Software Quality: A Measurement Framework for Assessment and Prediction*. Springer.

D. Nicolette (2009), *Agile Metrics*, http://www.slideshare.net/rsrivastava91/agile-metrics-v6

A. A. Noureddine, M. Damodaran, and S. Younes (2009), "A Framework for Harnessing the Best of Both Worlds in Software Project Management: Agile and Traditional," in *Information Systems Education Conference 2009*, Washington D.C.

J. A. O'Brien (2001), *Introduction to Information Systems: Essentials for the Internetworked E-Business Enterprise*. Boston: McGraw-Hill/Irwin.

C. A. O'Reilly Iii and M. L. Tushman (2004), "The Ambidextrous Organization," *Harvard Business Review,* vol. 82, pp. 74-81.

C. O'hEocha, K. Conboy, and X. Wang (2010), "So You Think You're Agile?," in *Agile Processes in Software Engineering and Extreme Programming*. vol. 48, A. Sillitti, A. Martin, X. Wang, and E. Whitworth, Eds., ed: Springer Berlin Heidelberg, pp. 315-324.

N. Oza, P. Abrahamsson, and K. Conboy (2009), "Positioning Agility," in *Agile Processes in Software Engineering and Extreme Programming*. vol. 31, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds., ed: Springer Berlin Heidelberg, pp. 206-208.

S. R. Palmer and J. M. Felsing (2002), *A Practical Guide to Feature Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR.

J. Patton (2008), "Ambiguous Business Value Harms Software Products," *IEEE Software,* vol. 25, pp. 50-51.

M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber (1993), "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute.

R. Petit (2006), Agile Processes: Making Metrics Simple, Agile Journal.

M. Poppendieck (2003), Lean Software Development. *C++ Magazine Methodology Issue*.

M. Poppendieck and T. Poppendieck (2003), *Lean Software Development: An Agile Toolkit*. Addison Wesley.

M. Poppendieck and T. D. Poppendieck (2010), *Leading Lean Software Development: Results Are Not the Point*. Upper Saddle River, NJ: Addison-Wesley.

R. S. Pressman (2005), *Software Engineering: A Practitioner's Approach*, 6 ed. McGraw - Hill International Edition.

A. Qumer and B. Henderson-Sellers (2008), "An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering," *Information and Software Technology,* vol. 50, pp. 280-295.

A. Qumer and B. Henderson-Sellers (2008), "A Framework to Support the Evaluation, Adoption and Improvement of Agile Methods in Practice," *Journal of Systems and Software,* vol. 81, pp. 1899-1919.

A. Qumer, B. Henderson-Sellers, and T. McBride (2007), "Agile Adoption and Improvement Model," in *European, Mediterranean & Middle Eastern Conference on Information Systems (EMCIS)*, Polytechnic University of Valencia, Spain.

D. F. Rico, H. H. Sayani, and S. Sone (2009), *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation*: J. Ross Publishing.

The Standish Group (1995), "The Chaos Report."

W. Royce (2002), CMM vs. CMMI: From Conventional to Modern Software Management. *Rational Edge*.

W. W. Royce (1970), “Managing the Development of Large Software Systems,” in *IEEE WESCON*, pp. 1-9.

W. W. Royce (1987), “Managing the Development of Large Software Systems: Concepts and Techniques,” in *Proceedings of the 9th international conference on Software Engineering*, Monterey, CA.

A. Ruiz (2010), *Effective Code Coverage (and Metrics in General)*, http://alexruiz.developerblogs.com/?p=1421

T. L. Saaty (1990), “How to Make a Decision: The Analytic Hierarchy Process,” *European Journal of Operational Research,* vol. 48.

C. Schwaber (2007), “The Truth About Agile Proceses - Frank Answers to Frequently Asked Questions,” Forrester Research.

K. Schwaber and J. Sutherland (2011), *The Scrum Guide*, http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf#zoom=100

M. Senapathi and A. Srinivasan (2012), “Understanding Post-Adoptive Agile Usage: An Exploratory Cross-Case Analysis,” *Journal of Systems and Software,* vol. 85, pp. 1255-1268.

J. Shore (2005), *Beyond Story Cards: Agile Requirements Collaboration*, http://jamesshore.com/Multimedia/Beyond-Story-Cards.html

J. Shore and S. Warden (2008), *The Art of Agile Development*. Sebastopol, CA: O'Reilly Media, Inc.

A. Sidky (2007), “A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework,” Dissertation, Computer Science, Virginia Tech, Blacksburg, VA.

A. Sidky and J. D. Arthur (2007), “Determining the Applicability of Agile Practices to Mission and Life-Critical Systems,” in *Proceedings of the 31st IEEE Software Engineering Workshop*.

A. Sidky and J. D. Arthur (2008), “Value-Driven Agile Adoption: Improving an Organization's Software Development Approach,” presented at the New Trends in Software Methodologies, Tools and Techniques - Proceedings of the Seventh SoMeT, Sharjah, United Arab Emirates.

A. Sidky, J. D. Arthur, and S. Bohner (2007), “A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework,” *Innovations in Systems and Software Engineering,* vol. 3, pp. 203-216.

J. M. Siviy, M. L. Penn, and R. W. Stoddard. (2008), *CMMI and Six Sigma: Partners in Process Improvement*. Upper Saddle River, NJ: Addison-Wesley.

G. Smith and A. Sidky (2009), *Becoming Agile in an Imperfect World*. Greenwich, CT: Manning Publications Co.

S. Soundararajan (2008), “Agile Requirements Generation Model: A Soft-Structured Approach to Agile Requirements Engineering,” Thesis, Computer Science, Virginia Tech, Blacksburg, VA.

S. Soundararajan (2011), “A Methodology for Assessing Agile Software Development Approaches,” *Computing Research Repository,* vol. abs/1108.0427.

S. Soundararajan and J. D. Arthur (2009), “A Soft-Structured Agile Framework for Larger Scale Systems Development,” in *16th Annual IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS)*, San Francisco, pp. 187-195.

S. Soundararajan and J. D. Arthur (2011), “A Structured Framework for Assessing the "Goodness" of Agile Methods,” in *18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, pp. 14-23.

S. Soundararajan, J. D. Arthur, and O. Balci (2012), "A Methodology for Assessing Agile Software Development Methods," in *Agile Conference (AGILE) 2012*, pp. 51 - 54.

S. Soundararajan, A. Chigani, and J. D. Arthur (2012), "Understanding the Tenets of Agile Software Engineering: Lecturing, Exploration and Critical Thinking," in *Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education*, Raleigh, North Carolina, USA, pp. 313-318.

M. L. Talbert (1995), "A Methodology for the Measurement and Evaluation of Complex System Designs," Dissertation, Computer Science, Virginia Tech, Blacksburg, VA.

Thoughtworks (2009), "Agile Self Evaluation Version 1.0 Prepared for Ciprian Mester, Alcatel-Lucent."

Thoughtworks (2013), *Agile Assessments*, http://www.agileassessments.com/

D. Ulrich and N. Smallwood (2004), "Capitalizing on Capabilities," *Harvard Business Review,* vol. 82, pp. 119-127.

F. van Latum, R. van Solingen, M. Oivo, B. Hoisl*, et al.* (1998), Adopting GQM-Based Measurement in an Industrial Environment. *IEEE Software*.

K. Waters (2008), *How Agile Are You? (Take This 42 Point Test)*, http://www.allaboutagile.com/how-agile-are-you-take-this-42-point-test/

L. Werth (1993), "Lecture Notes on Software Process Improvement," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Educational Materials CMU/SEI-93-EM-008. http://www.sei.cmu.edu/library/abstracts/reports/93em008.cfm

L. Williams and R. R. Kessler (2003), *Pair Programming Illuminated*. Boston: Addison-Wesley.

L. Williams, K. Rubin, and M. Cohn (2010), "Driving Process Improvement Via Comparative Agility Assessment," in *Agile Conference (AGILE) 2010,* pp. 3-10.

L. Williams, K. Rubin, and M. Cohn (2013), *Comparative Agility*, http://comparativeagility.com/

E. V. Wilson and S. D. Sheetz (2010), "A Demands-Resources Model of Work Pressure in It Student Task Groups," *Computers & Education,* vol. 55, pp. 415-426.

## Appendix A- Working definitions of the Objectives, Principles, and Strategies

**This section provides the working definitions for the objectives, principles, and strategies identified by the OPS Framework.**

## Objectives

***Human – centric***

People are more important than processes, practices and tools

***Value-driven***

Maximize stakeholder value(s): increased revenue, improved customer satisfaction, reduced cost, etcetera.

***Minimal Waste***

Keep things simple - build only what is necessary.

***Maximal Adaptability***

Maintain flexibility: (a) accommodate change and (b) freedom to choose appropriate practices

***Continuous innovation and learning***

Innovate and improve the development process through the frequent examination and evaluation of past development activities.

## Principles

***Frequent delivery of working software***

Deliver working software frequently; iteration length – 2 to 4 weeks.

***Technical Excellence***

Provide an environment for achieving technical excellence - select the right people, right process and right practices to build working software of value to the customer.

***Simplicity***

Keep the development process simple; produce a product that displays only the necessary functionality.

***Empowering teams of Motivated Individuals***

Build teams of motivated individuals and empower them; push the decision-making process to the lower (or lowest) level.

***Constant development Pace***

Build software at a constant pace - the amount of work performed during each iteration should be constant.

***Accommodating Change***

Accommodate change with minimal impact.

***Continual stakeholder communication and collaboration***

Promote interaction among the stakeholders at regular intervals.

***Frequent Reflection and Improvement***

Re-examine the development process regularly with the intent to better understand and improve that process.

***Striving for Customer Satisfaction***

Promote customer satisfaction; provide maximum value to the customer.

## Strategies

***Iterative progression***

Develop the product over several iterations/cycles in sequence. Decompose the overall development lifecycle into multiple timeboxed (fixed length) release cycles and each release cycle into timeboxed iterations.

***Incremental development***

Build the product incrementally. Develop only a selected/ prioritized set of features during a release cycle.

***Short delivery cycles***

Deliver valuable software frequently.

***Evolutionary requirements***

Allow the features/ requirements to evolve over the development lifecycle.

***Continuous feedback***

Gather feedback from the customers and users on a regular basis

***Distribution of expertise***

Select the right people to complete the tasks. Ensure that the team is composed of people with the appropriate skill sets to complete the assigned tasks.

***Test-first development***

Write the unit tests first before writing code. Also, capture the customer acceptance criteria for features and stories before proceeding to the downstream development activities.

***Refactoring***

Refine the architecture, design, code, and/or other process artifacts regularly to improve the quality of that artifact by altering its internal structure while preserving its external behavior.

***Adherence to standards***

Conform to a set of standards that the team or organization has agreed to comply with. The standards adopted are guided by the culture and values of the teams and organizations. E.g. Coding standards.

***Continuous integration***

"Team members integrate their work frequently; usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible."

***Configuration management***

"Manage the evolution of the product and other artifacts, both during the initial stages of development and during all stages of maintenance."

***Minimal documentation***

Maintain just-enough documentation to satisfy the needs of the development team and the customer.

***High bandwidth communication***

Facilitate continuous communication among the stakeholders (in-person, face-to-face interactions).

***Self-managing teams***

"Allow the team members to determine, plan, and manage their day-to-day activities and duties under reduced or no supervision."

***Constant velocity***

Maintain the amount of work done during each iteration at a constant.

***Retrospection***

Re-examine the development process regularly with the intent to better understand and improve the process

***Client-driven iterations***

"The choice of features for each release comes from the client – whatever they perceive as the highest business value to them." The customers and users prioritize the features. Build only what is of value to the customers and users.

## Appendix B - Linkages

## Substantiating Citations for Linkages between Objectives and Principles

| *Objectives/Principles* | *Freq. Delivery* | *Tech. Excellence* | *Simplicity* | *Emp. Teams of motivated individuals* | *Const. Development Pace* | *Acc. Change* | *Continual stkhlder comm. and coll.* | *Freq. refln and learning* | *Striving for cust. satisfaction* |
|---|---|---|---|---|---|---|---|---|---|
| Human-centric | | [4][7] | | [1][3][7] | [3] | | [3][4][7] | | [4] |
| Value-driven | [3] | [2][3] | | | | | [4][11][12] | | [4][10] |
| Minimal waste | [6][8] | | [6][8] | | | | | | [8][13] |
| Maximal Adaptability | [9] | | [5] | | | [3] | | [3] | |
| Continuous Innovation and Learning | | [3] | [3] | | | | | [3][4] | |

## Evidence for linkages between Objectives and Principles

[1] M. Mirakhorli, A. K. Rad, F. Shams, M. Pazoki*, et al.* (2008), "RDP Technique: A Practice to Customize XP," in *Proceedings of the 2008 International Workshop on Scrutinizing Agile Practices or Shoot-out at the Agile Corral (APOS '08)*, Leipzig, Germany, pp. 23-32.

[2] L. M. Maruping, V. Venkatesh, and R. Agarwal (2009), "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements," *Information Systems Research,* vol. 20, pp. 377-399.

[3] A. Koch (2004), *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House Publishers.

[4] J. Shore and S. Warden (2008), *The art of agile development*. Beijing; Sebastopol, CA: O'Reilly Media, Inc.

[5] G. Asproni (2004), "Motivation, Teamwork, and Agile Development," Agile Times, vol. 4, pp. 8-15.

[6] M. Poppendieck and T. Poppendieck (2003), Lean Software Development: An Agile Toolkit. Addison Wesley.

[7] M. Levy and O. Hazzan (2009), "Knowledge Management in Practice: The Case of Agile Software Development," in ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE '09), pp. 60-65.

[8] M. Poppendieck and T. D. Poppendieck (2010), *Leading Lean Software Development: Results Are Not the Point*. Upper Saddle River, NJ: Addison-Wesley.

[9] R. C. Martin (2003), *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR.

[10] M. Cohn (2006), *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

[11] B. Boehm and R. Turner (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.

[12] B. Boehm (2003), "Value-based Software Engineering," SIGSOFT Software Engineering Notes, vol. 28, p. 4.

[13] M. Poppendieck (2003), Lean Software Development. C++ Magazine Methodology Issue.

**Substantiating Citations for Linkages between Principles and Strategies**

| *Practices/Principles* | *Freq. Delivery* | *Tech. Excellence* | *Simplicity* | *Emp. Teams of motivated individuals* | *Const. Development Pace* | *Acc. Change* | *Continual stkhlder comm. and coll.* | *Freq. refln and learning* | *Striving for cust. satisfaction* |
|---|---|---|---|---|---|---|---|---|---|
| Iterative Progression | [1] | | | | [18] | [20] | | | |
| Incremental Devt. | [1] | | [1] | | | [20] | | | |
| Evolutionary Reqts. | [1] | | [13][14] | | | [21][13] | | | |
| Refactoring | | [4][5] | [14][15] | | | | | | |
| Test First Development | | [6][7][8][5][1] | [6][14][15] | | | | | | |
| Self-Managing teams | | | | [17] | | | | | |
| Constant Velocity | | | | | [19] | | | | |
| Continuous Feedback | [2] | | | | | [22][23] | [18] | [2][14] | [15] |
| Minimal Documentation | | | [16] | | | | | | |
| High-bandwidth communication | | | [14] | | | | [10][1] | [14] | |
| Retrospection | | | | | | [24] | | [25][26][27] | |
| Client-driven iterations | | | | | | | | | [28][15] |
| Short delivery cycles | [1][3] | | | | | | | [14] | [14][15] |
| Distribution of expertise | | [9][10][5][1] | | [9][10][14] | | | | | |
| Configuration Management | | [11] | | | | | | | |
| Adherence to standards | | [1][12] | | | | | | | |
| Continuous Integration | | [5] | | | | | | | |

## Evidence for linkages between Principles and Strategies

[1] A. Koch (2004), *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House Publishers.

[2] K. Petersen and C. Wohlin (2009), “A Comparison of Issues and Advantages in Agile and Incremental Development Between State of the Art and an Industrial Case,” *Journal of Systems and Software,* vol. 82, pp. 1479-1490.

[3] M. Mirakhorli, A. K. Rad, F. Shams, M. Pazoki*, et al.* (2008), “RDP Technique: A Practice to Customize XP,” in *Proceedings of the 2008 International Workshop on Scrutinizing Agile Practices or Shoot-out at the Agile Corral (APOS '08)*, Leipzig, Germany, pp. 23-32.

[4] P. Tingling and A. Saeed (2007), “Extreme Programming in Action: A Longitudinal Case Study,” in *Proceedings of the 12th international conference on Human-Computer Interaction: Interaction Design and Usability (HCI'07)*, Beijing, China, pp. 242-251.

[5] M. Poppendieck and T. D. Poppendieck (2010), *Leading Lean Software Development: Results Are Not The Point*. Upper Saddle River, NJ: Addison-Wesley.

[6] B. George and L. Williams (2004), “A Structured Experiment of Test-Driven Development,” *Information and Software Technology,* vol. 46, pp. 337-342.

[7] C. Desai, D. Janzen, and K. Savage (2008), “A Survey of Evidence for Test-Driven Development in Academia,” *SIGCSE Bulletin,* vol. 40, pp. 97-101.

[8] T. Bhat and N. Nagappan (2006), “Evaluating The Efficacy Of Test-Driven Development: Industrial Case Studies,” in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, Rio de Janeiro, Brazil, pp. 356-363.

[9] A. Cockburn (2002), *Agile Software Development*. Addison-Wesley.

[10] B. Boehm and R. Turner (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.

[11] A. M. J. Hass (2003), *Configuration Management Principles And Practice*. Boston, MA: Addison-Wesley.

[12] D. F. Rico, H. H. Sayani, and S. Sone (2009), *The Business Value of Agile Software Methods: Maximizing ROI With Just-in-time Processes and Documentation*. J. Ross Publishing.

[13] S. W. Ambler, *Agile Requirements Change Management*, http://www.agilemodeling.com/essays/changeManagement.htm

[14] J. Shore and S. Warden (2008), *The Art Of Agile Development*. Sebastopol, CA: O'Reilly Media, Inc.

[15] J. Highsmith (2002), *Agile Software Development Ecosystems*. Addison-Wesley.

[16] S. W. Ambler, *Agile/Lean Documentation: Strategies for Agile Software Development*, http://www.agilemodeling.com/essays/agileDocumentation.htm

[17] B. Appleton (2009), *Agile Self-Organizing Teams*, http://bradapp.blogspot.com/2009/06/agile-self-organizing-teams.html

[18] D. Karlstrom and P. Runeson (2005), “Combining Agile Methods with Stage-Gate Project Management,” *IEEE Software,* vol. 22, pp. 43-49.

[19] M. Bless (2011), *Agile Principle 8: Sustainable Pace*, http://marcbless.blogspot.com/2011/04/agile-principle-8-sustainable-pace.html

[20] M. Poppendieck and T. Poppendieck (2003), *Lean Software Development: An Agile Toolkit*: Addison Wesley.

[21] L. Cao and B. Ramesh (2008), Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*.

[22] T. Dingsøyr and G. K. Hanssen (2003), “Extending Agile Methods: Postmortem Reviews as Extended Feedback,” in *Advances in Learning Software Organizations*. vol. 2640, S. Henninger and F. Maurer, Eds., ed: Springer Berlin / Heidelberg, pp. 4-12.

[23] R. Vidgen and X. Wang (2009), “Coevolving Systems and the Organization of Agile Software

Development," *Information Systems Research,* vol. 20, pp. 355-376.

[24] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta (2002), *Agile Software Development Methods: Review and Analysis*. Finland: VTT Publications.

[25] D. J. Reifer (2002), "How to Get the Most out of Extreme Programming/Agile Methods," in *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, pp. 185-196.

[26] C. Keith (2008), *Retrospective Meetings*, http://blog.agilegamedevelopment.com/2008/06/when-we-talk-about-agile-we-use-phrase.html

[27] C. Larman (2004), *Agile and Iterative Development: A Manager's Guide*. Boston: Addison-Wesley.

## Appendix C – Indicator Hierarchies for Assessing Capability and Effectiveness

## Indicator Hierarchy – Capability

- ❖ Refactoring
  - ▪ Support for Refactoring
    - Is refactoring an expected activity?
    - Given the system's architecture and other development environment elements such as the state of the code base, is it feasible to implement code refactoring?
    - Given the system's architecture and other development environment elements such as the state of the code base, is it feasible to implement architecture refactoring?
  - ▪ Buy-in for Refactoring
    - Are the teams receptive to implementing refactoring?
    - Is the management receptive to supporting refactoring efforts?
  - ▪ Minimizing Technical Debt
    - Is it expected that a well-defined process be adopted to minimize technical debt?
    - Is it expected that a well-defined process be adopted to manage technical debt?
    - Is minimizing technical debt a high priority activity?

- ❖ Test First Development
  - ▪ Process Support for Test-First Development
    - Is test-first development an expected activity?
    - Are the customers expected to specify the acceptance criteria for the features and stories before the developers begin coding?
  - ▪ Tool Support for Test First Development
    - Do appropriate testing tools exist?
  - ▪ Unit Testing
    - Are the developers expected to write unit tests first for their code?

- ❖ Retrospection
  - ▪ Support for Retrospection
    - Is retrospection an expected activity?
  - ▪ Tool Support for Retrospection
    - Are tools available for recording the outcomes of the retrospective meetings?

- ❖ Distribution of expertise
  - ▪ Appropriate team composition
    - Is a scheme for appropriate team composition defined?
    - Are the requisite skillsets for particular projects identified upfront?
    - Is it expected that the right people be chosen to accomplish the tasks?

- ❖ Configuration Management
  - ▪ Tool Support for Configuration Management
    - Do tools for version control and management exist?
  - ▪ Support for Configuration Management
    - Is it expected that the code be kept up to date?
    - Is it expected that the tests be kept up to date?
    - Is it expected that the builds be kept up to date?
    - Is it expected that the release infrastructure be kept up to date?
    - Is it expected that the documentation be kept up to date?

- ❖ Adherence to standards
  - ▪ Identifying features
    - Is it expected that well-defined techniques be used to identify the features?
  - ▪ Estimation

        - Is it expected that a well-defined approach to estimating the amount of work to be done during each release cycle and iteration be used?
    - Requirements Prioritization
        - Is it expected that a well-defined approach to prioritizing features, stories, and tasks be used?
    - Feature Decomposition
        - Is it expected that a mechanism for decomposing the selected features to be developed during the current release cycle into stories be defined?
    - Coding standards
        - Is it expected that each team creates and adopts a set of coding standards?
        - Is it expected that practices such as pair-programming, collective code ownership be adopted or automated tools be used to ensure adherence to the set standards?

- Continuous Integration
    - Tool Support for Continuous Integration
        - Do automated test suites exist?
        - Does the requisite test environment exist?
        - Do appropriate configuration management systems exist?
    - Process Support for Continuous Integration
        - Is continuous integration an expected activity?
        - Are the team members expected to integrate their code every few hours?
        - Is it expected that the builds, tests, and other release infrastructure be kept up to date?
        - Is it expected that automated test suites be developed?
        - Is it expected that the build process be automated?
    - Buy-in for Continuous Integration
        - Are the teams receptive to implementaing continuous integration?
    - Story Completeness
        - Is it expected that the criteria for Done/Done be specified upfront?

- Self-managing teams
    - Team Empowerment
        - Are the team members expected to be involved in determining, planning, and managing their day-to-day activities?
    - Ownership
        - Are the team members expected to demonstrate individual or collective code ownership?
    - Performance Expectations
        - Is there a set of performance expectations that are agreed upon by the team and the management?

- Continuous Feedback
    - Customer Feedback
        - Does the process define a mechanism for the customers to provide feedback?
    - Customer Acceptance
        - Is it expected that the acceptance testing occur before the end of a release cycle?

- High-bandwidth communication
    - On-site Customer
        - Are the customers available onsite to answer questions and provide continuous feedback?
        - In the absence of an onsite customer, do the customers provide feedback via other means?
    - Scheduling
        - Is it expected that time be allocated for Release Planning?
        - Is it expected that time be allocated for Iteration Planning?
        - Is it expected that time be allocated for Retrospection?
        - Is it expected that time be allocated for Daily Progress Tracking meetings?
    - Inter- and intra-team communication
        - Is it expected that team members communicate and collaborate with their colleagues?

        - Do the teams have access to requisite tools to support inter- and intra-team communication?
    - Physical environment
        - Is the physical environment conducive to supporting high bandwidth communication?

- Client-driven Iterations
    - Identifying and prioritizing features
        - Are the customers expected to be involved in identifying the features?
        - Are the customers expected to establish the priorities of the features?

- Short delivery cycles
    - Development timeframes
        - Is it expected that the product be developed over short delivery cycles? For example, a product increment should be released every 6 - 12 months and iterations last for four weeks or less.

- Iterative Progression
    - Planning
        - Is the team expected to plan for each iteration?
    - Estimation Authority
        - Are the developers expected to estimate the time required to complete each story and feature?
    - Estimation
        - Is it expected that a well-defined approach to estimating the amount of work to be done during each release cycle and iteration be used?

- Incremental Development
    - Estimation Authority
        - Are the developers expected to estimate the time required to complete each story and feature?
    - Requirements Management
        - Are tools available for managing the features and stories?
    - Identifying and prioritizing features
        - Are the customers expected to be involved in identifying the features?
        - Are the customers expected to establish the priorities of the features?

- Evolutionary Requirements
    - Minimal BRUF and BDUF
        - Is it expected that only high level features be identified upfront?
        - Is it expected that an evolutionary approach to architecting the system be followed as opposed to creating the architecture upfront?
    - JIT Refinement
        - Is it expected that the requirements be determined and refined just-in-time?
    - Feature Decomposition
        - Is it expected that a mechanism for decomposing the selected features to be developed during the current relase cycle into stories be defined?

- Minimal Documentation
    - Tool Support for Minimal Documentation
        - Do tools for maintaining documentation exist?
    - Proces support for Minimal Documentation
        - Is it expected that minimal documentation be maintained?
    - Buy-in for Minimal Documentation
        - Are the teams receptive to maintaining minimal or just-enough documentation?

## Indicator Hierarchy – Effectiveness

- Refactoring
  - Minimizing Technical Debt
    - To what extent do the teams manage technical debt?
    - To what extent do the teams minimize technical debt when developing new systems?
    - To what extent does the system and the development environment allow Technical Debt to be minimized?
  - Buy-in for Refactoring
    - To what extent does the management support the implementation of refactoring?
    - To what extent do the teams implement refactoring?

- Test First Development
  - Code coverage
    - To what extent did the developers provide adequate code coverage from the tests?
  - Customer Satisfaction
    - To what extent is the product developed so far in-sync with the customers' needs and expectations?
  - Testing first
    - To what extent do developers write tests first before writing code?
    - To what extent are the test plans created before the developers start coding?

- Retrospection
  - Process Outcomes for Retrospection
    - To what extent were practices that worked well during the iteration or the release cycle and hence should be used in the future identified?
    - To what extent were practices that did not yield the expected results and hence should be discontinued identified?
    - To what extent were new practices that may better suit the team's needs identified?
  - Retrospective goals
    - To what extent were the retrospective goals set during the previous iteration met?

- Configuration Management
  - Project Environment for Configuration Management
    - To what extent do teams use appropriate tools for version control and management?

- Adherence to standards
  - Estimation
    - To what extent are the estimates for the amount of work to be done during each iteration accurate?
  - Coding standards
    - To what extent do the team members agree with the set coding standards?
    - To what extent do the team members adhere to the set coding standards?

- Continuous Integration
  - Project Environment for Continuous Integration
    - To what extent are automated test suites developed?
    - To what extent are the code bases not shared?
  - Story Completeness
    - To what extent has each story been coded?
    - To what extent has each story been unit tested?
    - To what extent has each story been refactored?
    - To what extent has each story been checked into the code base?
    - To what extent has each story been integrated with the existing code base?
    - To what extent has each story been reviewed?

    - To what extent has each story been accepted by the customer?
  - Daily/Frequent builds
    - To what extent do automated builds run one or more times everyday?

- Distribution of expertise
  - Process Outcomes for Distribution of Expertise
    - To what extent do the team members have the requisite expertise to complete the tasks assigned to them?
    - To what extent is the work assigned to the team members commensurate with their expertise?
    - To what extent does the team effectively complete the work that they have committed to?
    - To what extent do the teams have members in leadership positions that can guide the others?
    - To what extent do the teams not rely on knowledge external to their teams?

- Self-managing teams
  - Team Empowerment
    - To what extent do the team members determine the amount of work to be done?
    - To what extent do the team members take ownership of work items?
    - To what extent do the team members hold each other accountable for the work to be completed?
    - To what extent do the team members ensure that they complete the work that they are accountable for?
  - Autonomy
    - To what extent do the team members determine, plan, and manage their day-to-day activities under reduced or no supervision from the management?
    - To what extent do the developers form ad-hoc groups to determine and refine requirements just-in-time?
  - Management support
    - To what extent does the management support the self-managing nature of the teams?

- Continuous Feedback
  - Customer Feedback
    - To what extent do the customers provide feedback to the business and the development team?
  - Customer Satisfaction
    - To what extent is the product developed so far in-sync with the customers' needs and expectations?

- High-bandwidth communication
  - Customer Satisfaction
    - To what extent is the product developed so far in-sync with the customers' needs and expectations?
  - Scheduling
    - To what extent is the time allocated for the release planning meetings utilized effectively?
    - To what extent is the time allocated for the iteration planning meetings utilized effectively?
    - To what extent is the time allocated for the retrospective meetings utilized effectively?
    - To what extent is the time allocated for the daily progress tracking meetings utilized effectively?
    - To what extent do the scheduled meetings (except the daily progress tracking meetings) begin and end on time?
    - To what extent do the meetings (except the daily progress tracking meetings) take place as scheduled?
  - Inter- and intra-team communication
    - To what extent does open communication prevail between the business and the development team?
    - To what extent does open communication prevail between the manager/scrum master and the developers and testers?
    - To what extent does open communication prevail between the developers and the testers?
    - To what extent does open communication prevail among the developers?
    - To what extent does open communication prevail between the external customer/user and the business?
    - To what extent does open communication prevail between the external customer/user and the develpment team?
    - To what extent does open communication prevail between members of different teams?

- ❖ Client-driven Iterations
  - ▪ Requirements Prioritization
    - To what extent do the customers establish the priorities of the features?
  - ▪ Customer Satisfaction
    - To what extent is the product developed so far in-sync with the customers' needs and expectations?
  - ▪ Customer Requests
    - To what extent are the changes requested by the customers accommodated?

- ❖ Short delivery cycles
  - ▪ Development timeframes
    - To what extent is software released frequently? (length of a release cycle is one year or less)
    - To what extent is software released frequently? (length of an interation is four weeks or less)
  - ▪ Customer Satisfaction
    - To what extent is the product developed so far in-sync with the customers' needs and expectations?
  - ▪ Roll-backs
    - To what extent are the deployments not rolled back?

- ❖ Iterative Progression
  - ▪ Estimation
    - To what extent are the estimates for the amount of work to be done during each iteration accurate?
  - ▪ Iteration length
    - To what extent are the iterations timeboxed?
    - To what extent is the length of an iteration 4 weeks or less?
  - ▪ Requirements Management for Iterations
    - To what extent is an iteration backlog maintained?
    - To what extent are the stories fully estimated when added to the backlog?
    - To what extent are the stories prioritized when added to the backlog?

- ❖ Incremental Development
  - ▪ Requirements Management for Releases
    - To what extent is a product backlog maintained?
    - To what extent are the features priorotized when they are added to the backlog?
    - To what extent are the features fully estimated before they are added to the backlog?
  - ▪ Timeboxing Releases
    - To what extent are the release cycles timeboxed?
    - To what extent are only a subset of the identified features developed during a release cycle?

- ❖ Evolutionary Requirements
  - ▪ Requirements Reprioritization
    - To what extent are the features reprioritized as and when new features are identified?
  - ▪ Customer Requests
    - To what extent are the changes requested by the customers accommodated?
  - ▪ Minimal BRUF and BDUF
    - To what extent are only the high level features identified upfront?
    - To what extent are the architecture requirements allowed to evolve over time?

- ❖ Minimal Documentation
  - ▪ Maintaining documentation
    - To what extent is minimal documentation supported by teams?
    - To what extent is minimal documentation created/developed?
    - To what extent is minimal documentation recorded/archived?
    - To what extent is minimal documentation maintained?

## Appendix D – Survey Instruments used at the Agile Conference and at Company Z

## Survey Administered at the Agile 2012 Conference

## Assessment Questionnaire – A Methodology for Assessing Agile Software Development Approaches

| **SECTION 1: RESPONDENT INFORMATION** | |
|---|---|
| **Name (Optional):** | |
| **Affiliation (Optional):** | |
| **Position Title:** | |
| **Please rate your familiarity with Agile Software Development** | 1 2 3 4 5 6 7<br>Minimal Expert |
| **Number of years working within the Agile community** | 0 - 2 years 3 – 6 years 7 – 12 years >10 years |
| **In what capacity(ies) have you participated in Agile Software Development?**<br>__ Tester __ Developer __ Scrum Master/ Program Manager __ Business Analyst __ Coach<br>__ Other (Please specify): ______________________________________________ | |

| **SECTION 2: OBJECTIVES, PRINCIPLES, AND STRATEGIES (OPS) FRAMEWORK** | | | | | |
|---|---|---|---|---|---|
| **To what extent do you agree or disagree with the following statements:** | **Strongly Disagree** | **Slightly Disagree** | **Neither Disagree Nor Agree** | **Slightly Agree** | **Strongly Agree** |
| There a need for a systematic process for assessing the 'goodness' of an Agile Method. | | | | | |
| As a whole, the OPS Assessment Framework helps provide guidance to an organization in defining or refining its agile method. | | | | | |
| Adequacy, Capability, and Effectiveness are complementary views needed to support a comprehensive approach to assessing an Agile Method. | | | | | |
| The OPS Assessment Framework is a viable approach to assessing an agile method adopted by an organization. | | | | | |
| As a whole, the OPS Assessment Framework can help organizations evaluate their agile method(s) to identify the effective components, possible inadequacies, and potential enhancements. | | | | | |

**Do you have any other comments regarding the OPS Framework and its components?**

______________________________________________

______________________________________________

______________________________________________

______________________________________________

## Survey Administered at Company Z

## Assessment Questionnaire – A Methodology for Assessing Agile Software Development Approaches

| SECTION 1: RESPONDENT INFORMATION | |
|---|---|
| **Name (Optional):** | |
| **Affiliation (Optional):** | |
| **Position Title:** | |
| **Please rate your familiarity with Agile Software Development** | 1 2 3 4 5 6 7<br>Minimal Expert |
| **Number of years working within the agile community** | 0 - 2 years 3 – 6 years 7 – 10 years >10 years |
| **In what capacity(ies) have you participated in Agile Software Development?**<br>__ Tester __ Developer __ Scrum Master/ Program Manager __ Business Analyst __ Coach<br>__ Other (Please specify): ________________________________________________ | |

| SECTION 2: OBJECTIVES, PRINCIPLES, AND STRATEGIES (OPS) FRAMEWORK | | | | | |
|---|---|---|---|---|---|
| **RELEVANCE** | | | | | |
| **Is there a need for a systematic process to assessing the "goodness" of an Agile Methodology? If not, explain.** | | | | | |
| **To what extent do you agree or disagree with the following statements:** | **Strongly Disagree** | **Slightly Disagree** | **Neither Disagree Nor Agree** | **Slightly Agree** | **Strongly Agree** |
| The identified objectives, principles, and strategies are reflective of those observed in Agile Software Development processes. | | | | | |
| As a whole, the OPS Assessment Framework helps provide guidance to an organization in defining a more customized approach better suited for its needs. | | | | | |
| **COMPLETENESS** | | | | | |
| **To what extent do you agree or disagree with the following statements:** | **Strongly Disagree** | **Slightly Disagree** | **Neither Disagree Nor Agree** | **Slightly Agree** | **Strongly Agree** |
| Adequacy, Capability, and Effectiveness are complementary views needed to support a comprehensive approach to assessing an Agile Methodology. | | | | | |
| The following components of the OPS Framework are integral to forming a viable assessment approach:<br><br>(a) The identified sets of objectives, principles, and strategies | | | | | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| (b) The identified linkages among the objectives, principles, and strategies |  |  |  |  |  |
| (c) The hierarchical set of measurement indicators |  |  |  |  |  |
| (d) The OPS Framework that integrates the above three components |  |  |  |  |  |
| **UTILITY** |  |  |  |  |  |
| **To what extent do you agree or disagree with the following statement:** | **Strongly Disagree** | **Slightly Disagree** | **Neither Disagree Nor Agree** | **Slightly Agree** | **Strongly Agree** |
| The OPS Assessment Framework supports a viable approach to assessing:<br><br>(a) The adequacy of an Agile Methodology |  |  |  |  |  |
| (b) The capability of an organization to implement its selected Agile Development Methodology |  |  |  |  |  |
| (c) The effectiveness of an Agile Methodology |  |  |  |  |  |

## SECTION 3: GENERAL QUESTIONS

**Would you add, remove, or redefine any of the identified objectives, principles, and strategies? If so, please elaborate.**

**Are we targeting the right assessment indicators, i.e., people, process, project, and product? If not, what are we missing?**

**We assess an agile method from three perspectives: Adequacy, Capability, and Effectiveness. Are there other perspectives that we should consider?**

**Do you have any other comments regarding the OPS Framework and its components?**