CHALMERS |  UNIVERSITY OF GOTHENBURG

Measuring Agility

Master's Thesis in Software Engineering

KONSTANTINOS CHRONIS

Division of Software Engineering
Department of Computer Science & Engineering
Gothenburg University
Chalmers University of Technology
Gothenburg, Sweden 2014
Master's Thesis 2014:6

Abstract

Context:

Objective:

Method:

Results:

Acknowledgements

Konstantinos Chronis, Gothenburb, Sweden November 25, 2014

Contents

List of Tables

List of Figures

1

Introduction

Agile and plan-driven methodologies are the two dominant approaches in the software development. The eternal battle between them seems will go on for a long time before one of them prevails. Organizations and companies have the tendency to leave the cumbersome area of Waterfall process and to embrace the Agile methodologies. Although it has been almost 20 years since the latter were introduced, companies are quite reluctanct in following them. Once they do, they start enjoying the agile benefits, but are these the only benefits they could enjoy?

In order to answer to the previous question one should first understand what does "*agile*" mean? According to a dictionary, it means "*to be able to move quickly and easily*", something which is almost impossible with a plan-driven approach. The term agility was first introduced as agile manufacturing in an industry book [33].

In 2001, 17 developers formed the Agile Alliance and created the agile manifesto [7] defining what is considered to be agile in order to avoid confusion:

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

Software development teams started to adopt the most known agile methodologies, such as eXtreme Programming [5], Feature Driven Development (FDD) [36], Crystal [11], Scrum [46] and others. Most companies use a tailored methodology by following some processes and practices of the aforementioned ones in order to better suit their needs. Williams et al. [60] reports that rarely all XP practices are exercised in their pure form something to which Reifer [43] and Aveling [4] also agree based on the results of their surveys showing that it is common for organizations to partially adopt XP. Sidky

et al. [49] mention four issues organisations face when transitioning to agile. a) the organization's readiness for agility b) the practices it should adopt c) the potential difficulties in adopting them d) the necessary organizational preparations for the adoption of agile practices. The most important issue though that tends to be neglected, is how well are these methodologies adopted?

According to Escobar-Sarmiento and Linares-Vasquez [17] the agile methodologies are easier to misunderstand. Such a case could lead to problems later on in the software development process. The aforementioned statement is also supported by Taromirad and Ramsin [54] as well who argue that the agile software development methdologies are often applied to the wrong context. Livermore [32] does not only see eye to eye with the previously mentioned researchers but also comes to the conclusion that organizations modify practices before implementing them which is also mentioned by Patel et al. [37]. Hossain et al. [21] argues that improper use of agile practices creates problems. Sahota [45] states that doing agile and being agile are two different things. For the first one a company should follow practices while for the other one a company should think in an agile way. Lappo and Andrew [30] state that organizations following the practices of a methodology does not mean they gain much in terms of agility, while on the other hand Sidky [48] defines the level of agility of a company as the amount of agile practices used. Considering the aforementioned statement, a group that uses pair programming and collective code ownership at a very low level is more agile than a group which uses only pair programming but in a more efficient manner.

Williams et al. [61] pose the question "*How agile is agile enough*"? Practictioners think that declaring being agile is equally good as being agile. According to a survey from Ambysoft [2] only 65% of the agile companies that answered met the five agile criteria posed in the survey. In addition, 9% of agile projects failed due to lack of cultural transition while 13% of companies are at odds with core agile values based on the most recent survey by VersionOne [58]. Poonacha and Bhattacharya [38] mentioned that the different perception of agile practices when they are adopted is very worrying, since even people in the same team understand them differently according to the result of a survey [1]. It is evident not only from literature but also from its application that agile is a way of thinking and working, it is a whole culture [38]. If we had to use one word we could state it is a way of *being*. Nietzsche [35] said "*better know nothing than half-know many things*". In the same context maybe it is better not to transition to agile instead of thinking of being agile.

Since agile methodologies become more and more popular there is a great need for development of a tool that can measure the level of agility in the organizations that have adopted them. Sidky et al. [49] mentions of success stories about companies that have adopted agile methods, but without having a measurement tool that could tell if you are really agile. Reasearchers for more than a decade have been constantly coming up with models and frameworks in an effort to provide a solution. Unfortunately the multiple tools have created a saturation in the field resulting in being used only from the organizations that participated in the empirical studies for their creation [24][25]. As a result, the vicious circle of creating tools with no actual use holds back not only

the software development companies, but the research community as well.

1.1 Thesis Structure

- Introduction - page 1
- Related Work - page 4
- Case Study - page 14
- Results
- Discussion
- Appendix - page 38

2

Related Work

2.1 Introduction

According to Yauch [62] it is very difficult to measure agility, although it has been widely spread. Tsourveloudis and Valavanis [57] agree on this mainly because of the vagueness of agility's concept. Nevertheless, various tools have been developed in the last decade in order to measure the agility in software development teams. Below is a short description of some of the ones that have been used as references in most papers of this field. The tools are separated in two categories, a) those which measure the agility of the methodologies and b) those which measure the agility of organisations/teams.

2.2 Agility of Methodologies

2.2.1 Balancing Discipline and Agility

Boehm and Turner [8] did not come up with a tool to measure agility but rather to balance between agility and discipline. According to them [9] discipline is the foundation for any successful endeavour and creates experience, history and well-organized memories. On the other hand agility is described as a counterpart of discipline. Agility uses the memory and history in order to adjust in the context which is applied and takes advantage of the unexpected opportunities that might come up. The combination of the two can bring success to an organisation. Boehm and Turner [8] in their research came up with five "critical decision factors" which can determine if an agile or plan-driven method is suitable for a software development project.

Figure 2.1 depicts these factors: a) size of a team working in a project b) criticality of damage of unexpected defects c) culture needed to balance between chaos and order d) dynamism of the team working in chaos or in a planned way e) personnel which refers to the extended Cockburn [10] skill rating

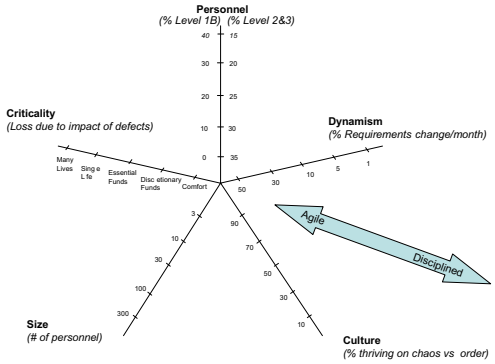

Figure 2.1: Dimensions affecting method selection

If the ratings of the five factors are close to the center, then the team is to an agile territory and the team is considered agile, otherwise it follows a discipline approach.

2.2.2 Philip Taylor - Assessing Tool

Taylor et al. [55] modified the tool created by Boehm and Turner [8] by adding a sixth axis named *Client Involvement* which has the following categories:

- On AB - Client is on-site and an agile believer. This is the ideal when a client is fully persuaded of the agile approach and makes themselves available onsite to work with the team.
- Off AB - Client is off-site but an agile believer. Although off-site, the client fully understands the nature of agile development and is open to frequent communication.
- On AS - Client is on-site but is an agile skeptic. They may be on-site but they are not convinced about the agile development approach.
- Off AS - Same as On AS except the problem is compounded by the client being off-site.
- Off Uninvolved - Not only is the client off-site but they want no involvement between providing the initial requirements and getting the right product delivered.

2.2.3 Datta - Agility Measurement Index

Datta [14] presented a metric to help in deciding which agile methodology best suits a project. The metric identifes five dimensions: a) Duration b) Risk c) Novelty d) Effort e) Interaction. For each one of these dimensions the user assigns a value, then by the use of a formula the user can identify whether Waterfall, Unified Process or eXtreme Programming is more appropriate.

2.2.4 Comprehensive Evaluation Framework for Agile Methodologies

Taromirad and Ramsin [54] created the "Comprehensive Evaluation Framework for Agile Methodologies" (CEFAM) in order to provide coverage to the important aspects of agile methodology. The tool consists of a hierarchy of evaluation criteria which are divided into five groups (see Figure 2.2) a) Process b) Modeling Language c) Agility d) Usage e) Cross-Context. Each of these groups has a number of questions which are either answered with a numeric value, with Yes/No or any value from a proposed set. At the end the answers are evaluated based on the following scale: Unacceptable ≤ 0.25; $0.25 <$ Low ≤ 0.5; $0.5 <$ Medium ≤ 0.75; $0.75 <$ High ≤ 1.0.

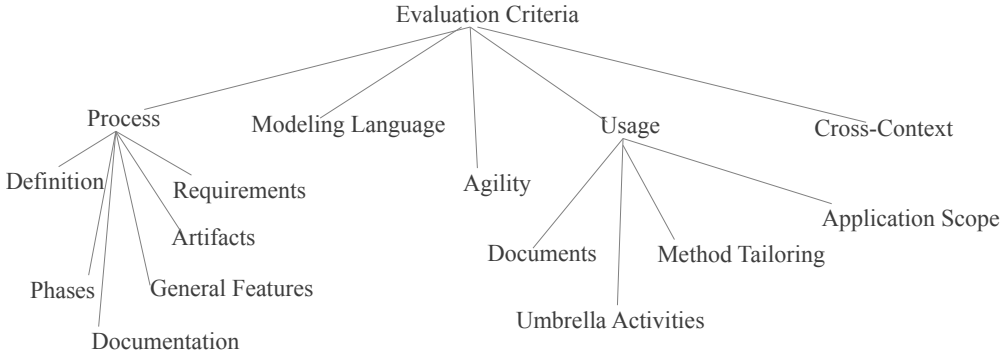

Figure 2.2: Evaluation criteria hierarchy for CEFAM

2.2.5 4-Dimensional Analytical Tool

Qumer and Henderson-Sellers [39] created the 4-Dimensional Analytical Tool (4-DAT) for analysing and comparing agile methods which is part of the of the Agile Adoption and Improvement Model (AAIM) [40]. The objective of the tool is to provide a mechanism to assess the degree of agility and adaptability of any agile methodology. The measurements are taken at a specific level in a process and use specific practices.

Dimension 1 - Method Scope Characterization The first dimension describes the key scope items which are considered essential for supporting the method used by a team or organisation. They have been derived from their literature review based on Beck and Andres [5], Koch [29], Palmer and Felsing [36], Highsmith [20] and provides a method comparison at a high level.

Dimension 2 - Agility Characterization The second dimension is the only quantitative dimension of the four. It evaluates the agile methods in process level and in a method practices level in order to check the existence of agility.

The measurement of the degree of agility in this level is done based on five variables.

These variables are used to check the existence of a method's objective at a specific level or phase. If the variable exists for a phase then the value 1 is assigned to it, oth-

erwise 0. Qumer and Henderson-Sellers [39] define the degree of agility (DA) as "the fraction of the five agility variables that are encompassed and supported".

Formula (2.1) how DA is calculated

$$DA(Object) = (1/m) \sum mDA(Object, PhaseOrPractices) \tag{2.1}$$

3 - Agile Values Characterization The third dimension consists of six agile values. Four of them are derived directly from the Agile Manifesto [7], while the fifth comes from Koch [29]. The last value is suggested by Qumer and Henderson-Sellers [39] after having studied several agile methods.

Dimension 4 - Software Process Characterization The fourth dimension examines the practices that support four processes as these are presented by Qumer and Henderson-Sellers [39].

Table 2.1: 4-DAT Dimensions

D1 - Scope	a) Project Size b) Team Size c) Development Style d) Code Style e) Technology Environment Responsiveness f) Physical Environment g) Business Culture h) Abstraction Mechanism
D2 - Features	a) Flexibility b) Speed c) Leanness d) Learning e) Responsiveness
D3 - Agile values	a) Individuals and interactions over processes and tools b) Working software over comprehensive documentation c) Customer collaboration over contract negotiation d) Responding to change over following a plan e) Keeping the process agile f) Keeping the process cost effective
D4 - Process	a) Development Process b) Project Management Process c) Software Configuration Control Process / Support Process d) Process Management Process

2.2.6 XP Evaluation Framework

Williams et al. [60] proposed a framework named the "XP Evaluation Framework" (XP-EF) for assessing the XP practices which have been adopted by an organization. The framework consists of three parts

- XP Context Factors (XP-CF) - Record a important contextual information. The factors can be team size, project size, staff experience

- XP Adherence Metrics (XP-AM) - Epxress in a precise way the practices utilized by a team
- XP Outcome Measures (XP-OM) - A Means to assess the outcomes of a project using full or partial XP practices

2.3 Agility of Organisations

2.3.1 Team Agility Assessment

Leffingwell [31] created a model for assessing the team's agility by taking into account six aspects: a) Product Ownership b) Release Planning and Tracking c) Iteration Planning and Tracking d) Team e) Testing Practices f) Development Practices/Infrastructure

Each of these aspects is followed by a number of questions rated on a six-point Likert scale and the results are represented in a radar chart.

2.3.2 Comparative Agility

Williams et al. [61] created the Comparative Agility (CA) assessment tool which does not assess the agility of an organization by providing an absolute value, but it rather provides a value in comparison to other organizations/companies [12]. The idea behind CA is that organizations try to be more agile than their competitors because of believing to have more benefits. Until 2010 more than 1200 respondents supported that idea by answering the tool's online survey. The CA assessment tool consists of the following seven dimensions a) Teamwork b) Requirements c) Planning d) Technical Practices e) Quality f) Culture g) Knowledge-Creating which are made up of three to six characteristics. Each characteristic has four statements where each one of them represents an agile practice. The answers of every statement are on a five-point Likert scale.

2.3.3 Escobar - Vasquez Model for Assessing Agility

Escobar-Sarmiento and Linares-Vasquez [17] created their own agility assessment model which consists of four stages. For the first three they use the models and tools proposed by other researchers they found in literature.

- Agile Project Management Assessment - proposed by Qumer and Henderson-Sellers [39]
- Project Agility Assessment - proposed by Taylor et al. [55]
- Workteam Agility Assessment - proposed by Leffingwell [31]
- Agile Workspace Coverage

For collecting the data for the measurements they used surveys based on the tools of each stage while in the last one they use their custom survey. The data are then

depicted in a four axis radar chart in order to provide a view of the company's agility. In Figure 2.3 one can see the model with a short description about which tool should be used at each level for each stage.

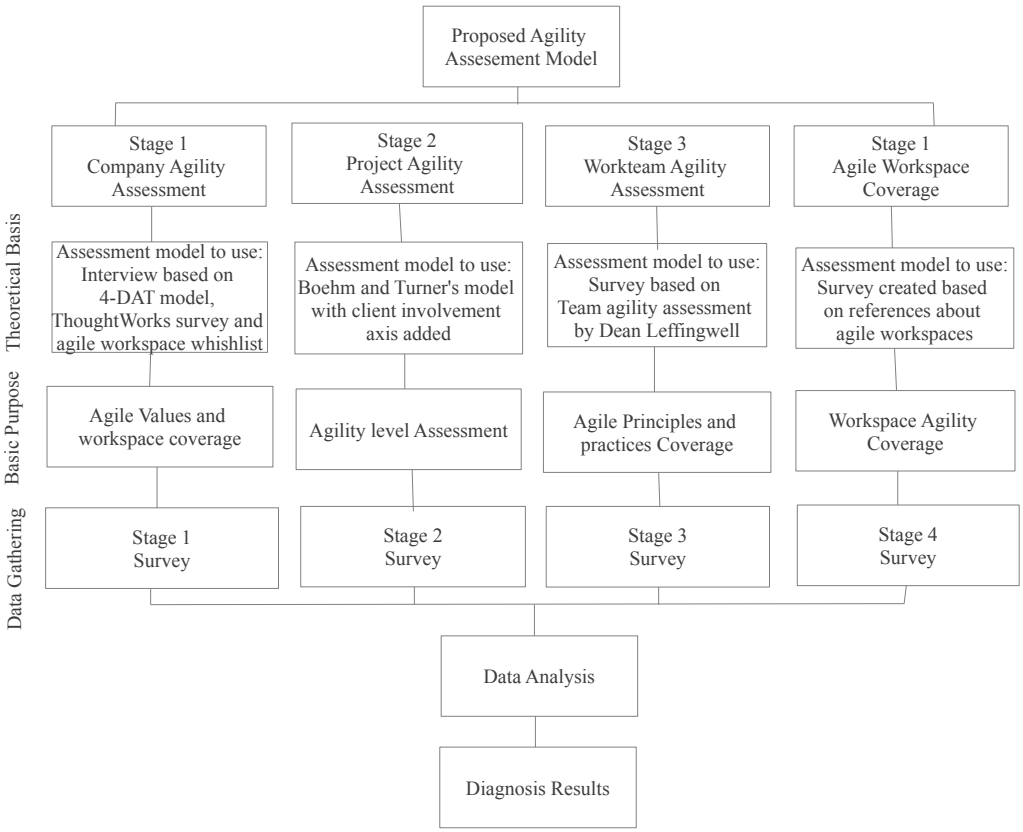

Figure 2.3: Escobar - Vasquez model for assessing agility

2.3.4 Shawky and Ali - Entropy Analysis

Shawky and Ali [47] measure the agility based on the rate of entropy change over the time of a system's development. If the rate is high then the process is of high agility as well. Each feature is considered to be an entity and the change logs of the entities are analyzed. They define as $P_i(t)$ the probability of an entity i to be associated with the change logs at a time t. Then by using formula (2.2) they make the calculation for the agility measure *AM(t)* for that specific time.

$$AM(t) = -\sum_{i=1}^{n} P_i'(t)(log_2 P_i(t) + 1.44) \tag{2.2}$$

2.3.5 Validation Model to Measure the Agility

Ikoma et al. [23] measure agility by creating a validation model since according to them only validation can confirm the quality of a product. In this model any candidate for validation item enters an "identified planning state" at planning time. Afterwards these items change to the "unvalidated inventory state" when the items start to be generated. Finally, validation of the deliverable items changes the state to the "validated product state" (see Figure 2.4). Then based on the formula (2.3) one can get the result. A is the agility of a project/organization, V' is the number of software items in the "validated product state" and U' is the average number of software items in which intermediate deliverables are in the "unvalidated inventory state".

$$A = V'/U' \tag{2.3}$$

2.3.6 Perceptive Agile Measurement

So and Scholl [50] created a survey for measuring agility from a social-psychological perspective, covering eight agile practices which they named as scales. These scales are an attempt to establish a representative set of agile practices commonly used in the field a) Iteration Planning b) Iterative Development c) Continuous Integration and Testing d) Co-Location e) Stand-up Meetings f) Customer Access g) Customer Acceptance Tests h) Retrospectives. The survey is on a seven-point Likert scale, except from the *Co-Location* which is on a five-point scale.

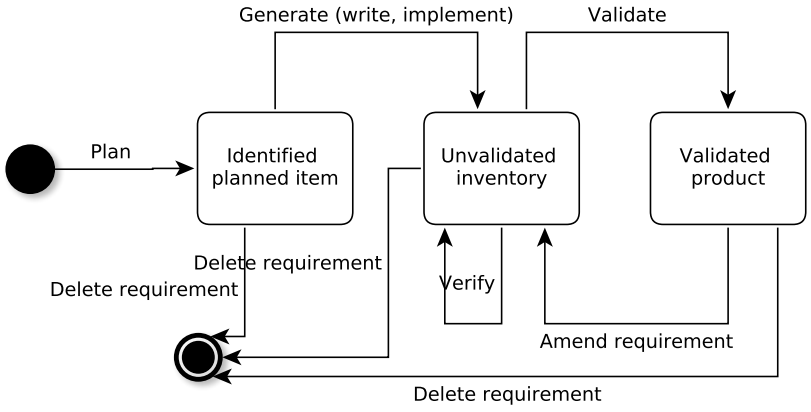

Figure 2.4: Validation Model to Measure the Agility

2.3.7 AHP - ANFIS Framework

Poonacha and Bhattacharya [38] created tool for measuring agility by identifing 17 parameters grouped in four parameter groups as it can be seen in Table 2.2. While the last group is an indicator of performance, the first three groups migitate the risks of supply, operation and demand uncertainties respectively. Each parameter is given as a

question and the answers were fed in the Adaptive Network based Fuzzy Inference Systems (ANFIS). Due to the complexity of the ANFIS model an Analytical Hierarchical Process (AHP) is mandatory to minize it.

Table 2.2: AHP - ANFIS Framework parameters

Group	**Parameters**
People	a) Attrition b) Functional Flexibility c) Training and Knowlegde d) Decentralized Decision Making e) Bench Strength
Processes	a) Pair Programming and Parallel Testing b) Iterative Development c) Degree of modularity d) Requirement Capture Process e) Reusability f) Continuous Improvement
Customer Involvment	a) Customer Involvement in Design b) Team Across Company Borders c) Customer Training Period
Cost and Quality	a) Cost of Requirement change b) Projects dropped due to incapacity c) Software Quality

2.3.8 Thoughtworks

Thoughtworks [56] is a worldwide consulting company. They have developed an online survey for assessing agility based on 20 multiple choice questions. The questions cover the areas of a) Requirements Analysis b) Business Responsiveness c) Collaboration and Communication d) Project Management e) Governance. People can answer to the survey online and they will get a report evaluating at which level their team or company is. The survey gained a lot of fame due to Martin Fowler [18], one of the creators of the agile manifesto working at the company.

2.3.9 42-Point Test

Waters [59] createad a simple 42-question survey based on a similar one from Nokia [53], in order to allow to Scrum/XP teams to easily see to what extent they follow various agile practices.

2.3.10 Objectives Principles Strategies Framework

Soundararajan [51] created the Objectives, Principles and Stategies (OPS) Framework in order to assess the "goodness" of an agile methodology. It is an evolution of the work done by Arthur and Nance [3] and Sidky [48]. The focus of this tool is mainly on eXtreme

Programming, Feature Driven Development, Lean, Crystal and any tailored instances of them.

In order to achieve this the framework examines the methodology based on 3 aspects:

- 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.

The OPS framework identifies a) objectives of the agile philosophy b) principles that support the objectives c) strategies that implement the principles d) linkages that relate objectives to principles, and principles to strategies e) indicators for assessing the extent to which an organization supports the implementation and effectiveness of those strategies

In total 5 objectives, 9 principles, 17 strategies 54 linkages and 80 indicators are identified. Also check Figure 2.5.

Agile Values	Objectives	Principles	Strategies
Individuals and Interactions	Human Centric	Frequent Delivery of Working Software	Iterative Progression
			Incremental Development
		Technical Excellence	Short Delivery Cycles
			Evolutionary Requirements
	Value-driven	Simplicity	Continuous Feedback
			Refactoring
Working Software	Minimal Waste	Empowering Teams of Motivated Individuals	Test First Development
			Self-Managing Teams
		Constant Development Pace	Continuous Integration
			Constant Velocity
	Maximal Adaptability	Accommodating Change	Minimal Documentation
			High-bandwidth Communication
Customer Collaboration		Continual Stakeholder Communication and Collaboration	Retrospection
			Client-driven iterations
	Continuous Innovation And Learning	Frequent Reflection and Improvement	Appropriate distribution of expertise
			Configuration Management
Responding to Change		Striving For Customer Satisfaction	Adherence to Standards

Figure 2.5: Objectives, Principles, and Strategies identified by the OPS Framework

2.3.11 Sidky Agile Measurement Index

Sidky [48] created the Sidky Agile Measurement Index (SAMI) in order to measure agility as part of the "Agile Adoption Framework". SAMI is a scale used by an agile coach to identify the potential of a project or organization [49], which consists of five agile levels and five agile principles, deriving from the agile manifesto [7], forming a 5x5 matrix. In most of the cells of this matrix there have been assigned agile practices. The assessement of agility takes place at each level by measuring the practices adopted by a team. Before moving to the next level the team needs to fulfill all the practices of the current one.

Agile Levels

- Level 1 - Collaborative
- Level 2 - Evolutionary
- Level 3 - Effective
- Level 4 - Adaptive
- Level 5 - Ambient

Agile Principles

- Embrace Change to Deliver Customer Value
- Plan and Deliver Software Frequently
- Human Centric
- Technical Excellence
- Customer Collaboration

3

Case Study

3.1 Problem

3.1.1 Research Question

- In what ways do the tools correlate?

3.2 Company F Information

3.2.1 Company F

Company F [1] is a United States company which acts in the POS [2] area. With the development of some new products the company had a 400% increase in the size of the development and QA departments resulting in the need for organizing better the development and release processes. In addition the increasing requests of new features in the company's systems requires a more efficient way in delivering them to the customers and also maintaining the quality of the products.

3.2.2 Methodology F

In general, company F does not follow a specific agile methodology, but rather a tailored mix of the most famous ones which suits the needs of each team. Methodology F, as we can name it, embraces the practices displayed in Table 3.1 from the various agile methodologies, some of the them in bigger and some of them in a smaller extent. For identifying these methodologies the analysis made by Koch [29] was used. The results were verified by the agile coach.

[1] F is the first letter of the company's name
[2] Point Of Sales

Table 3.1: Practices embraced by methodology F

Method	Practice
XP	a) Small Releases b) Simple design c) Refactoring d) Collective ownership e) Continuous integration f) 40-hour week g) Coding standards
FDD	a) Developing by feature b) Feature teams c) Regular build schedule d) Inspections e) Configuration management
Lean	a) Empower the team b) Build Integrity In c) Amplify learning d) Eliminate waste

3.2.3 Products

Company F has developed a few products which belong in the following four areas a) desktop b) mobile c) cloud d) platforms. The names given are respectives to the name of the teams that develop them.

- Product A - A series of three mobile applications which offer services to stores or customers of stores.
- Product B - A cloud application which offers services to product A and product D.
- Product C - A platform used only by the company's employees. It supports services though which are necessary for product D.
- Product D - It is the main product of the company which is mostly used. The rest of the products were developed in order to support it and expand its functionalities.

3.2.4 Teams

There are four development teams, each for a product of the company. Some of the teams have mixed members of developers and testers. In the Tables 3.2, 3.3, 3.4, 3.5, one can see the structure of the teams.

Table 3.2: Team A - Profile

Team Size	7
Roles	Team Leader (1) Developers (3) Testers (3)
Area	Mobile
Tools Used	Perforce Titanium
Iteration Length	2-3 weeks

Table 3.3: Team B - Profile

Team Size	6
Roles	Team Leader (1) Developers (5) Testers (1)
Area	Java
Tools Used	Perforce Eclipse IDE
Iteration Length	2-3 weeks

Table 3.4: Team C - Profile

Team Size	4
Roles	Team Leader (1) Developers (2) Testers (1)
Area	Java
Tools Used	Perforce Eclipse IDE
Iteration Length	3-4 weeks

Table 3.5: Team D - Profile

Team Size	19
Roles	Team Leader (1) Developers (10) Testers (8)
Area	Java
Tools Used	Perforce Eclipse IDE
Iteration Length	2-4 weeks

3.3 Methodology

In order to see which of the OPS, PAM and TAA can better measure the agility of software development teams, the four teams in company F were used as a sample.

3.3.1 Data Collection

Surveys

In order to collect the data an online survey was considered to be the best option since it could be easily answered by each subject and in addition this would assure no data loss. Google Drive ™[16] was selected to be the platform for collecting and preserving the data.

For each of the tools, four surveys were created for each team respectively. The collection lasted about one month, the surveys for each tool every ten days. First PAM was sent, then TAA and at last it was OPS.

Two subjects were requested to answer to the surveys first, in order to detect if there were any questions which could cause confusion, but also to see how much time is needed to fill in the surveys. Once the issues pointed out by the two subjects were fixed, the surveys were sent to the rest of the company's employees.

The links to the surveys were sent to the subjects early in the morning with an email, but they were asked to answer to them after lunch. The reasoning for this, is that at the beginning of the day the employees need to perform tasks which are usually important and time consuming and require to have clear thoughts and be on meetings. On the contrary, after lunch most of the employees try to relax by enjoying their coffee and discussing with each other. That time of the day was considered to be the best in order to ask them to spend 15-20 minutes and reply to the survey. The employees that belonged to more than one team, were asked a couple of days later to take the other survey in order to be more detached in their answers. Every question of the surveys was mandatory.

As it was mentioned in chapter 2, PAM focuses on the following agile practices: a) Iteration Planning b) Iterative Development c) Continuous Integration And Testing d) Stand-Up Meetings e) Customer Access f) Customer Acceptance Tests g) Retrospectives h) Co-Location. From these methodology F does not support *Stand-Up Meetings* and *Retrospectives* and as a result they were totally excluded from the surveys.

On the contrary, TAA focuses on the following agile practices/areas: a) Product Ownership b) Release Planning and Tracking c) Iteration Planning and Tracking d) Team e) Testing Practices f) Development Practices/Infrastructure. From the above practices, methodology F does not support *Product Ownership*, since it implies that company F should practice Scrum, which it does not. Moreover, Scrum oriented questions from the rest of the practices/areas were removed as well.

Finally, OPS focuses on the following strategies: a) Iterative progression b) Incremental development c) Short delivery cycles d) Evolutionary requirements e) Continuous feedback f) Refactoring g) Test-first development h) Self-managing teams i) Continuous integration j) Constant velocity k) Minimal documentation l) High bandwidth communication m) Retrospection n) Client-driven iterations o) Distribution of expertise p) Configuration management q) Adherence to standards

From the above practices, methodology F does not support *Retrospection*. The company considers that meetings with the whole team does not the desired results but rather the opposite leading in more difficult communication and loss of time. In case a retrospection on the past events has to be held, this takes place in meeting of two or three people.

As it was stated earlier, in subsection 2.3.10, OPS measures agility based on three aspects. Adequacy, Capability and Effectiveness. Effectiveness measurement focuses on how well a team implements agile methodologies. Since the rest of the tools focus on the same thing, it was decided only to use the survey from Effectiveness and not take into account about Adequacy and Capability.

For a more clear view on the questions contained in the surveys, one can take a look at Appendices A, B and C.

Likert Scales

The surveys for PAM TAA and OPS were on a Likert scale 1-7 (never-always). From PAM, only the *Co-location* practice had its Likert scale 1-5 (different time zones-same room) since its creators preferred it in this way. For the transforamtion of the results for this practice, formula (3.1) [22] was used.

$$x_2 = (1.5 * x_1) - 0.5 \tag{3.1}$$

Demographics

The employees that were asked to answer to the surveys where all the members of the software development teams which consisted of software and quality assurance (QA) engineers. All of the participating employees have more than a year in the company, while most of them have more than five years of work experience in an agile environment. Employees which had been working for less than six months in the company were not asked since it was considered they were not fully aware of the company's procedures or they were not familiar enough with them. Although code review is practised, it was avoided to ask from the code reviewers to take the same survey for the same team from that perspective because it would not provide more value to the results and in addition it would tire them a lot.

3.3.2 Collected Data

Each respondent replied to 176 questions in total. Initially, 34 surveys were expected to be filled in, but at then end 30 of them were filled in, since some employees preferred not to participate in the study.

3.3.3 Data Analysis

As it was mentioned earlier (if not, mention why and how) the data gathered from the survey were grouped by the practices covered by OPP and as a consequence OPS. From the 18 practices in total, four of them, a) Minimal or Just Enough Documentation b) Customer User Acceptance Testing c) Evolutionary Requirements d) Constant Velocity , are covered only by one tool. The rest of the practices were covered by at least two of the tools.

Initially there was a thought to analyze the data for each team separately but this would create an extensive overhead. In addition, teamA has so few members that the results would be inadequate to work with. As a result, it was prefered to form the data sets for each practice based on the answers from all the teams. In Table 3.6 one can see the structured of the collected data.

Table 3.6: Collected Data Structure

Practice	**Participants**	**PAM**	**TAA**	**OPS**
Practice1	Participant1	Score1	Score1	Score1
	⋮	⋮	⋮	⋮
	ParticipantN	ScoreN	ScoreN	ScoreN

Based on the analysis performed at chapter 4 it is clear that OPS covers more agile practices/areas. As a result it was decided to use these practices in order to check the correlation of the tools. In Appendix D one can see how the questions are group based on the OPS practices. The data sets consist of the answers which were sorted by team as stated above and have the same order in all practices (i.e. the nth answer in every practice is given by the same person).

For the mathematical calculations the RStudio ™[44] was selected since it has a wide support from its community.

For the analysis of the data the correlation method was selected as it has been used in other cases of tools comparisons [26] [15]. The initial thought was to use "Pearson product-moment correlation" as it is the one mostly used in statistics.

The four practices mentioned above were discarded for using correlation, since there would not be any.

In order to use Pearson's correlation there must hold four prerequisites

1. The two variables should be measured at the interval or ratio level
2. The variables should be approximately normally distributed
3. There needs to be a linear relationship between the two variables
4. There should be no significant outliers

Since the data were collected by surveys using Likert scales, then prerequisite #1 is satisfied, as the Likert scales can be considered both for interval or ratio level measurements, depending on the problem each time.

As far as the normal distribution is concerned, the Shapiro-Wilk test was selected as it appears to be the most powerful normality test according to a recent paper published by Razali and Wah [42]. In order for a distribution to be considered normal, the p-value must be greater than the alpha level so as not to reject the null hypothesis and consider that the data are normally distributed. The chosen alpha level was 0.05 as it is the most common one.

Out of the 42 normality checks (three for each of the 14 practices) only 17 concluded that the data are normally distributed.

The low level of normally distributed data gave a strong indication that the Pearson's correlation would not be the appropriate way to measure the tools' correlation. As a result it was preferred not to continue checking the last two prerequisites for Pearson's

correlation, but rather use "Spearman's rank correlation coefficient" which is for non-parametric data.

In order to use Spearman's rank correlation coefficient two prerequisites must be satisfied

1. The two variables should be measured at the interval or ratio level
2. There needs to be a monotonic relationship between the two variables

The first prerequisite was already covered above. In order to check for the monotonicity, plots were drawn between the results of each tool for all 14 practices. The plots surprisingly showed that only 8 out of 42 were monotonic, which does not allow to use Spearman's correlation for the rest. The table 3.7 summarizes the practices and the relationships in which Spearman's correlation can be used.

Table 3.7: Pairs of tools for which Spearman's correlation can be used

Practice	**Tools Correlation**
Continuous Feedback	PAM - OPS
High Bandwidth Communication	PAM - TAA PAM - OPS TAA - OPS
Client Driven Iterations	PAM - OPS
Continuous Integration	PAM - TAA
Iterative and Incremental Development	PAM - OPS
Refactoring	PAM - TAA

The cells for which a correlation exists are coloured in green in order to distinguish from the rest. In Table 3.14 one can see that half of the correlations are between PAM and OPS. This corroborates that OPS has full coverage of PAM as seen in Table 4.6.

Table 3.8: Continuous Feedback Correlations

Continuous Feedback			
	PAM	TAA	OPS
PAM	1.000	NA	0.459
TAA	NA	1.000	NA
OPS	0.459	NA	1.000

Table 3.9: Client Driven Iterations Correlations

Client Driven Iterations			
	PAM	TAA	OPS
PAM	1.000	NA	0.161
TAA	NA	1.000	NA
OPS	0.161	NA	1.000

Table 3.12: Continuous Integration Correlations

Continuous Integration			
	PAM	TAA	OPS
PAM	1.000	0.398	0.249
TAA	0.398	1.000	0.115
OPS	0.249	0.115	1.000

Table 3.13: Iterative and Incremental Development Correlations

Iterative and Incremental Development			
	PAM	TAA	OPS
PAM	1.000	0.204	0.396
TAA	0.204	1.000	-0.228
OPS	0.396	-0.228	1.000

Table 3.10: High Bandwidth Communication Correlations

High Bandwidth Communication			
	PAM	TAA	OPS
PAM	1.000	0.322	-0.023
TAA	0.322	1.000	0.237
OPS	-0.023	0.237	1.000

Table 3.11: Refactoring Correlations

Refactoring			
	PAM	TAA	OPS
PAM	1.000	0.097	-0.050
TAA	0.097	1.000	0.181
OPS	-0.050	0.181	1.000

Table 3.14: Frequency of correlation between tools

Frequency	
PAM-OPS	4
PAM-TAA	3
TAA-OPS	1

In Table 3.15 one can see the descriptive statistics of the data gathered.

Table 3.15: Descriptive Statistics

Practice	Statistics	PAM	TAA	OPS	Practice	Statistics	PAM	TAA	OPS
Adherence to Standards	Mean	1.00	11.67	8.10	Appropriate Distribution of Expertise	Mean	1.00	11.13	27.20
	Sd	0.00	2.17	2.12		Sd	0.00	2.10	3.51
	Median	1	12	8		Median	1.0	11.5	27.0
	Min	1	7	6		Min	1	6	21
	Max	1	14	12		Max	1	14	35

Client-Driven Iterations	Mean	8.63	1.00	13.87	Continuous Feedback	Mean	4.87	1.00	9.20
	Sd	3.20	0.00	2.78		Sd	1.25	0.00	1.88
	Median	8.5	1.0	14.0		Median	5.0	1.0	9.5
	Min	3	1	9		Min	2	1	5
	Max	14	1	21		Max	7	1	14
Continuous Integration	Mean	21.97	24.13	48.10	High-Bandwidth Communication	Mean	36.73	22.87	60.30
	Sd	4.40	3.82	4.23		Sd	4.11	3.25	5.69
	Median	21.0	24.5	48.5		Median	38	23	60
	Min	11	16	40		Min	29	13	51
	Max	31	31	56		Max	42	28	75
Iteration Progress Tracking and Reporting	Mean	21.67	71.73	31.73	Iterative and Incremental Development	Mean	27.10	8.43	14.47
	Sd	6.42	15.62	1.55		Sd	2.71	2.11	2.13
	Median	22.5	72.5	32.0		Median	27.0	8.5	15.0
	Min	8	40	27		Min	22	4	11
	Max	35	100	35		Max	34	13	18
Product Backlog	Mean	1.00	4.97	15.80	Refactoring	Mean	2.03	10.80	20.67
	Sd	0.00	0.85	2.14		Sd	0.85	2.27	3.66
	Median	1.0	5.0	15.5		Median	2.0	11.0	20.5
	Min	1	3	12		Min	1	6	14
	Max	1	6	19		Max	4	14	28
Self-Organizing Teams	Mean	3.6	62.9	36.5	Smaller and Frequent Product Releases	Mean	5.6	5.8	24.8
	Sd	1.19	6.57	5.20		Sd	1.19	0.81	1.24
	Median	3.5	63.0	37.0		Median	6	6	25
	Min	2	48	26		Min	2	4	22
	Max	6	75	45		Max	7	7	28

Software Configuration Management	Mean	1	7	7	Test Driven Development	Mean	10.90	6.57	9.10
	Sd	0	0	0		Sd	2.90	3.28	1.97
	Median	1	7	7		Median	10.5	6.0	9.0
	Min	1	7	7		Min	6	3	6
	Max	1	7	7		Max	17	15	13
Minimal or Just Enough Documentation	Mean	1.0	1.0	17.8	Customer User Acceptance Testing	Mean	17.37	1.00	1.00
	Sd	0.00	0.00	3.16		Sd	7.04	0.00	0.00
	Median	1	1	18		Median	17.5	1.0	1.0
	Min	1	1	10		Min	5	1	1
	Max	1	1	23		Max	33	1	1
Evolutionary Requirements	Mean	1.00	1.00	20.13	Constant Velocity	Mean	1.00	5.93	1.00
	Sd	0.00	0.00	2.21		Sd	0.00	1.01	0.00
	Median	1	1	20		Median	1	6	1
	Min	1	1	17		Min	1	4	1
	Max	1	1	25		Max	1	7	1

3.4 Threats to Validity

3.4.1 Construct Validity

3.4.2 Internal Validity

According to Gren [19] internal validity is possibly the greatest threat when measuring agility. The creators of PAM, TAA and OPS though have already tried to mitigate this when creating their tools.

3.4.3 External Validity

Every software development team even in the same company applies agile methodologies in different ways. Nevertheless the correlations among the tools shouldn't differ a lot due to the reasons explained in section 3.5. Yet, as it is known, conducting a case study is susceptible in generalizing its outcomes. In this case though we can consider that teams following the same agile practices shouldn't have different results.

3.5 Analysis

3.5.1 Introduction

As it was seen by the reader the correlation of only a few of the practices can be examined. In this section the reasons that did not allow the calculation of the correlation will be presented and discussed.

3.5.2 Correlations Analysis

In *Continuous Feedback* PAM and OPS have a moderate positive correlation of ρ = 0.459. Both tools focus on getting feedback from the customer, while OPS also checks whether the product is developed according to the customer's needs and expectations.

In *Client-Driven Iterations* PAM and OPS have a low positive correlation of ρ = 0.161. Both tools check for the possibility of the requirements having been prioritized by the customer. Moreover OPS focuses on the customers' requests and needs.

In *Continuous Integration* PAM and OPS have a low positive correlation of ρ = 0.249. The common areas are continuous builds, multiple submits and story acceptance. There is a small difference on whether the developers should sync to the latest available code, with PAM supporting it.

Iterative and Incremental Development PAM and OPS have a low positive correlation of $\rho = 0.396$. OPS focuses on the stories estimation and prioritization while PAM on the deadlines having to meet and on the software progress.

High Bandwidth Communication PAM and TAA have a low positive correlation of ρ = 0.322. Both of them check about the team co-location while TAA also checks about the communication with the customers. PAM and OPS have correlation of ρ = -0.023 which means there is no correlation at all. They both focus on the communication, but OPS does that to huge extent, resulting in the so extremely low correlation. In addition, OPS checks about effectively using the time for meetings. TAA and OPS have a positive correlation of $\rho = 0.237$. This is the only practice for which correlation exists for all the tools.

Refactoring PAM and TAA have a correlation of $\rho = 0.097$, which means there is almost no correlation at all. TAA focuses on continuous refactoring while on the other hand PAM on unit testing.

3.5.3 Reasons

The tools measure a practice in the same way

Tools that claim to measure agility in the same way are expected to do so. Unfortunately this does not imply that their correlation can be calculated. The figure F.13 made it very clear why. All the answers from all the teams and for TAA and OPS were the same. Both tools had a unique question about using a version control management system, which stands true for company F for anything that has to do with code changes. The designed plots only had one spot which means the correlation among the tools is

not possible to calculate due to lack of variation in the results. Such an occurrence shows that correlation techniques can be defective when checking tools which manage to measure the same thing to the same extent.

The tools have few or no questions for measuring a practice

Another reason for not being able to calculate the correlation of the tools is that they cover slightly or not even at all some of the practices. An example of this, is the *Smaller and Frequent Product Releases* practice. OPS has four questions for it, while on the other hand PAM and TAA have a single one each. Furthermore, *Appropriate Distribution of Expertise* is not covered at all by PAM while it is by the rest of the tools. In case the single question gets a low score this will affect the effectiveness of the tool. On the contrary, multiple questions can cover more completely the practice by examining more factors that affect it. Apart from measuring a practice more precisely this also has the benefit that even if a question gets a low score, the rest of them are candidates for getting a higher one. The issue is worse if a practice is not covered at all. Unfortunately this phenomenon does not allow to equally check all the tools giving a respected superiority to the ones that take the practice into consideration and even more to the ones that do that to an extended degree.

The tools measure the same practice differently

Something very interesting that came up during the data analysis was that the tools although they cover the same practices they do it in different ways, leading in different results. An example of this is the practice of *Refactoring* F.10. PAM checks whether there are enough unit tests and automated system tests to allow the safe code refactoring. In case of course unit/system tests are not developed by a team, then the respondents will give low scores to the question, as the team members in company F did. Nevertheless, this does not mean that the team never refactors the software or it does it with bad results. All teams in company F choose to refactor when it adds value to the system, but the level of unit tests is very low and they exist only for specific teams. On the other hand, TAA and OPS check how often do the teams refactor among other factors which allows to better evaluate the practice of refactoring. Considering the above, PAM seems to fail to measure the *Refactoring*.

The tools measure the same practice in opposite questions

The *Continuous Integration* practice has a unique paradox among TAA, PAM and OPS. The first two tools have a question about the members of the team having synced to the latest code, while OPS checks for the exact opposite. According to Soundararajan [51] it is preferable for the teams not to share the same code in order to measure the practice. It is quite doubtful though how correct can this question be, since the *Continuous Integration* requires for frequent submits from the developers and thus the rest of the team will also have a local version of the code.

Questions phrasing

Although the tools might cover the same areas for each practice, the results could differ because of how a question is structured. An example of this is the *Test Driven Development* practice. Both TAA and PAM ask about automated code coverage, while OPS just asks about the existence of code coverage. Furthermore, TAA focuses on 100% automation while PAM doesn't. Thus, if a team has code coverage but it is not automated, then the score to the respective question should be low. In case of TAA if it is fully automated it should be even lower. It is evident that the more specific a question is, the more its answer will differ resulting in possible low scores.

How people perceive agility — Maybe add this at the final discussion of the document?

Although the concept of agility is not new, people don't seem to fully understand it. This is actually the reason for having so many tools in the field trying to measure how agile are the teams or the methodologies used by them. Teams implement agile methodologies differently and researchers create different measurement tools. There are numerous definitions about what agility is [27, 28, 34, 41], and each of the tools creator adopt or adapt one to their needs. Their only common basis is the agile manifesto [7] and its twelve principles [6] which are (and should be considered as) a compass for the agile practitioners. Nevertheless, they are not enough resulting in the saturation of the field. Moreover, Conboy and Fitzgerald [13] state that the Agile Manifesto principles do not provide practical understanding of the concept of Agility. Consequently, all the aforementioned reasons for the survey results as stated above, are driven by the fact how people perceive agility. The word people refers to the tool creators and tool users.

The questions in the surveys were all based on how their creators perceived the agile concept which as explained above it is vague. As the reader has seen in previous chapters PAM, TAA and OPS focus on some common areas/practices, such as *Smaller and Frequent Product Releases* and *High-Bandwidth Communication* while many are different. None of the Soundararajan [51], So and Scholl [50], Leffingwell [31] claimed of course to have created the most complete measurement tool, but still this leads to the oxymore that tools created by specialists to measure the agility of software development tools, they actually do it differently without providing substantial solution to the problem. On the contrary, this leads to more confusion to agile practitioners who are at a loose ends.

Considering that the researchers and specialists in the agile field perceive differently the concept of agility it would be naive to say that teams don't do the same. The answers in surveys are subjective and people answer them depending on how they understand them. This is also corroborated, by the fact that although a team works at the same room and follows the same processes for weeks it is rather unlikely if its members will have the same understanding what it means for them a retrospection or a releasing planning meeting.

3.6 Discussion

This chapter tried to check the correlations between *Perceptive Agile Measurement*, *Team Agility Assessment* and *Objectives Practices and Strategies* tools. One would expect that their results would be similar, since their creators claim they can measure the agility of software development teams. Unfortunately, as it was seen in the analysis, the correlations among them were only few and most of them were very low (check subsection 3.5.2).

4

Tools Completeness

4.1 Introduction

4.1.1 Research Question

- How much complete are the tools in measuring agility?

4.2 Team Agility Assessment Areas

Team Agility Assessment (TAA) does not state covering specific agile practices, but rather areas important for a team. It focuses on product ownership for Scrum teams but also on the release and iteration planning and tracking. The team factor plays a great role but also the development practices and the working environment. Automated testing is important here as well. Finally it is worth mentioning that it is the only tool focusing so much on the release planning. In Table 4.1 one can see TAA's areas.

Table 4.1: Areas covered by TAA

TAA Areas	
• Product Ownership	• Team
• Release Planning and Tracking	• Testing Practices
• Iteration Planning and Tracking	• Development Practices / Infrastructure

4.3 Perceptive Agile Measurement Practices

The Perceptive Agile Measurement (PAM) tool focuses on the iterations during software development but also on the stand-up meetings among the team members, their collocation and the retrospectives they have. The customers access and their acceptance criteria have a high significance as well. Finally the continuous integration and the automated unit testing are considered crucial in order to be agile. In Table 4.2 one can see PAM's practices.

Table 4.2: Agile practices covered by PAM

PAM Practices	
• Iteration Planning	• Stand-up Meetings
• Iterative Development	• Customer Access
• Continuous Integration and Testing	• Customer Acceptance Tests
• Co-Location	• Retrospectives

4.4 Objectives, Principles, Strategies Practices

Objectives, Principles, Strategies (OPS) Framework is the successor of the Objectives, Principles, Practices (OPP) Framework [52]. OPP identified 27 practices as implementations of the principles which later on were transformed into 17 strategies. In Table 4.3 one can see OPP's practices.

4.5 Practices Covered Between The Tools

As it can be clearly seen between Tables 4.3, 4.2 4.1 OPP and as a consequence OPS covers more agile practices than the other tools.

In the next pages follows a mapping between OPP and PAM (see Table 4.4) and OPP and TAA (see Table 4.5).

Some of the OPP practices though have abstracted to OPS strategies in order to avoid repetition of the questions' mapping and in order to better reflect the OPS Framework. The OPP practices a) *Frequent Face-to-Face Communication* b) *Physical Setup Reflecting Agile Philosophy* c) *Collocated Customers* have been abstracted to the OPS strategy *High-Bandwidth Communication* [51, p. 57]. In the same way the OPP *Automated test*

Table 4.3: Agile practices covered by OPP

OPP Practices	
• Iterative and Incremental Development	• Appropriate Distribution of Expertise
• Continuous Feedback	• Self-Organizing Teams
• Evolutionary Requirements	• Client-Driven Iterations
• Smaller and Frequent Product Releases	• Product Backlog
• Customer/User Acceptance Testing	• Agile Project Estimation
• Frequent Face-to-Face Communication	• Adherence to Coding Standards
• Refactoring	• Physical Setup Reflecting Agile Philosophy
• Automated Test Builds	• Daily Progress Tracking Meetings
• Software Configuration Management	• Minimal or Just Enough Documentation
• Test Driven Development	• Minimal Big Requirements Up Front and Big Design Up Front
• Iteration Progress Tracking and Reporting	• Collocated Customers
• Code Ownership	• Constant Velocity
• Retrospectives Meetings	• Pair Programming
• Just-in-Time Refinement of Features /Stories/Tasks	

builds practice has been abstracted to the OPS strategy *Continuous Integration* [51, p. 57].

The connection between the practices and strategies is done based on the questions of each tool. The aforementioned connections are depicted with colours. When a practice has more than one colour, it is because it covers more practices from the other tool (The colours and symbols among Tables 4.4, 4.5 are randomly selected and do not imply any connection between the two tables).

Table 4.4: Relation of OPP/OPS and PAM practices

PAM	OPP/OPS
Iteration Planning ✦	Iteration Progress Tracking and Reporting ✦
Iterative Development ✤	Iterative and Incremental Development ✦ ✤
Continuous Integration and Testing ❇	Continuous Integration ❇
Co-Location ✻	Software Configuration Management ❇
Stand-up Meetings ✷	Test Driven Development ❇ ✳
Customer Access ✣	High-Bandwidth Communication ✣ ✻
Customer Acceptance Tests ✳	Daily Progress Tracking Meetings ✷
Retrospectives ✠	Client-Driven Iterations ✣ ✦
	Evolutionary Requirements ✳
	Customer/User Acceptance Testing ✳
	Retrospectives Meetings ✠
	Self-Organizing Teams ✦

4.6 Mapping of questions from the PAM and TAA tools

PAM has divided its questions based on agile practices, while on the other hand TAA has divided them based on areas considered important. In the next pages there is a mapping of the questions used from the PAM and TAA tools, with the practices from OPP and strategies from OPS. As one can see from the tables above, while all practices/areas from PAM and TAA are mapped to OPP and OPS, not all of their questions are under OPP practices or OPS strategies. This can be explained due to the different perception/angle, of the creators of the tools, of what is important for an organization to be agile.

The questions among the tools will be match based on whether they are covered directly, relevantly, or not at all. Direct match will be considered the one where a question from a tool is the same or almost similar with one from OPS. Relevant match will be considered the one where a question of a tool does not exist in OPS, but its practice does exist in OPS. Non relevant match will be considered the one where a question cannot be matched at all in OPS.

The detailed mapping of the tools can be viewed in Appendix D.

Table 4.5: Relation of OPP/OPS and TAA practices/areas

TAA	OPP/OPS
Product Ownership ✻	Iterative and Incremental Development ✻
Release Planning and Tracking ✷	Product Backlog ✻
Iteration Planning and Tracking ✦	Smaller and Frequent Product Releases ✷
Team ✠	Customer/User Acceptance Testing ✦ ✷
Testing Practices ❃	Constant Velocity ✦
Development Practices/Infrastructure ✣	Iteration Progress Tracking and Reporting ✦
	Self- Organizing Teams ✠ ✷ ✦ ✣
	Appropriate Distribution of Expertise ✠
	High-Bandwidth Communication ✠
	Daily Progress Tracking Meetings ✠
	Retrospectives Meetings ✠ ✦ ✷
	Test Driven Development ❃
	Refactoring ✣
	Software Configuration Management ✣
	Adherence to Coding Standards ✣
	Pair Programming ✣
	Continuous Integration ✣ ❃

4.7 Analysis

By viewing Tables 4.4 4.5, and Appendix D one can clearly distinguish that OPP and consequently OPS is more complete than the others in measuring agility, covering all the areas of the PAM and TAA tools. Furthermore, as it can be seen in Table 4.6 OPS covers a high percent of questions from both tools directly and relevantly. TAA has a respective percent of non relevant matches mostly due to *Product Ownership* perspectives which is not covered in such an extent from OPS. This can be explained by the fact that OPS covers basis methodologies for developing software such as XP, FDD, Crystal, Lean [51, p. 44] whereas *Product Ownership* refers explicitly to Scrum which is a method for managing product development [29].

Table 4.6: Questions Coverage from OPS

Questions Coverage		
Match	**PAM**	**TAA**
Direct Match	17/48 (35.4%)	25/68 (36.7%)
Relevant Match	31/48 (64.5%)	33/68 (48.5%)
Non Relevant	0/48 (0%)	10/68 (14.7%)

5

Unknown

5.1 Introduction

- Can the tools be combined in a way so that they can produce a more complete approach on measuring agility?

6

Discussion

7

Future Work

Appendices

A

Objectives Principles Strategies - Effectiveness

The items marked with ✗ are the ones not included in the surveys to the teams.

- 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?

- 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?

- 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?

- 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?

- 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 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 development 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 story?
  - 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 time-frames
    - * 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 iteration 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 Big Requirements Up Front and Big Design Up Front
        * 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?

B

Perceptive Agile Measurement

The items marked with ✗ are the ones not included in the surveys to the teams.

- Iteration Planning
  - All members of the technical team actively participated during iteration planning meetings
  - All technical team members took part in defining the effort estimates for requirements of the current iteration
  - When effort estimates differed, the technical team members discussed their underlying assumption
  - All concerns from team members about reaching the iteration goals were considered
  - The effort estimates for the iteration scope items were modified only by the technical team members
  - Each developer signed up for tasks on a completely voluntary basis
  - The customer picked the priority of the requirements in the iteration plan
- Iterative Development
  - We implemented our code in short iterations
  - The team rather reduced the scope than delayed the deadline
  - When the scope could not be implemented due to constraints, the team held active discussions on re-prioritization with the customer on what to finish within the iteration
  - We kept the iteration deadlines
  - At the end of an iteration, we delivered a potentially shippable product

  - The software delivered at iteration end always met quality requirements of production code
  - Working software was the primary measure for project progress

- Continuous Integration And Testing
  - The team integrated continuously
  - Developers had the most recent version of code available
  - Code was checked in quickly to avoid code synchronization/integration hassles
  - The implemented code was written to pass the test case
  - New code was written with unit tests covering its main functionality
  - Automated unit tests sufficiently covered all critical parts of the production code
  - For detecting bugs, test reports from automated unit tests were systematically used to capture the bugs
  - All unit tests were run and passed when a task was finished and before checking in and integrating
  - There were enough unit tests and automated system tests to allow developers to safely change any code

✗ Stand-Up Meetings
  - ✗ Stand up meetings were extremely short (max. 15 minutes)
  - ✗ Stand up meetings were to the point, focusing only on what had been done and needed to be done on that day
  - ✗ All relevant technical issues or organizational impediments came up in the stand up meetings
  - ✗ Stand up meetings provided the quickest way to notify other team members about problems
  - ✗ When people reported problems in the stand up meetings, team members offered to help instantly

- Customer Access
  - The customer was reachable
  - The developers could contact the customer directly or through a customer contact person without any bureaucratical hurdles
  - The developers had responses from the customer in a timely manner
  - The feedback from the customer was clear and clarified his requirements or open issues to the developers

- Customer Acceptance Tests

  - How often did you apply customer acceptance tests?
  - A requirement was not regarded as finished until its acceptance tests (with the customer) had passed
  - Customer acceptance tests were used as the ultimate way to verify system functionality and customer requirements
  - The customer provided a comprehensive set of test criteria for customer acceptance
  - The customer focused primarily on customer acceptance tests to determine what had been accomplished at the end of an iteration

✗ Retrospectives

  ✗ How often did you apply retrospectives?
  ✗ All team members actively participated in gathering lessons learned in the retrospectives
  ✗ The retrospectives helped us become aware of what we did well in the past iteration(s)
  ✗ The retrospectives helped us become aware of what we should improve in the upcoming iteration(s)
  ✗ In the retrospectives (or shortly afterwards), we systematically assigned all important points for improvement to responsible individuals
  ✗ Our team followed up intensively on the progress of each improvement point elaborated in a retrospective

- Co-Location
  - Developers were located majorly in Greece
  - All members of the technical team (including QA engineers, db admins) were located in Greece
  - Requirements engineers were located with developers in Greece
  - The project/release manager worked with the developers in Greece
  - The customer was located with the developers in Greece

C

Team Agility Assessment

The items marked with ✗ are the ones not included in the surveys to the teams.

- ✗ Product Ownership
  - ✗ Backlog prioritized and ranked by business value
  - ✗ Backlog estimated at gross level
  - ✗ Product owner defines acceptance criteria for stories
  - ✗ Product owner and stakeholders participate at iteration and release planning
  - ✗ Product owner and stakeholders participate at iteration and release review
  - ✗ Product owner collaboration with team is continuous
  - ✗ Stories sufficiently elaborated prior to planning meetings
- Release Planning and Tracking
  - Release theme established and communicated
  - Release planning meeting attended and effective
  - Release backlog defined
  - Release backlog ranked by priority
  - Release backlog estimated at plan level
  - The team has small and frequent releases
  - The team has a common language and metaphor to describe the release
  - Release progress tracked by feature acceptance
  - ✗ Team completes and product owner accepts the release by the release date
  - ✗ Release review meeting attended and effective

  - Team inspects and adapts (continuous improvement) the release plan
  - Team meets its commitments to release

- Iteration Planning and Tracking
  - Iteration theme established and communicated
  - Iteration planning meeting attended and effective
  - Team velocity measured and used for planning
  - Iteration backlog defined
  - Iteration backlog ranked by priority
  - Team develops and manages iteration backlog
  - Team defines, estimates, and selects their own work (stories and tasks)
  - Team discusses acceptance criteria during iteration planning
  - Team manages interdependencies and constraints
  - Iteration progress tracked by task to do (burn-down chart) and card acceptance (velocity)
  - ✗ Work is not added by the product owner during the iteration
  - ✗ Team completes and product owner accepts the iteration
  - Iterations are of a consistent fixed length
  - Iterations are no more than four weeks in length
  - ✗ Iteration review meeting attended and effective
  - Team inspects and adapts (continuous improvement) the Iteration Plan

- Team
  - The whole team is present at release planning meetings
  - Team is cross-functional with integrated product owner, development, documentation and QA
  - Team is colocated
  - Team is 100% dedicated to the release (no time-slicing)
  - Team is smaller than 15 people
  - Team works in a physical environment that fosters collaboration
  - Team works at a sustainable pace
  - Team members complete commitments
  - ✗ Daily standup on time, fully attended and effectively communicates
  - Team leads communication; communication not managed
  - Team self-polices and reinforces use of agile practices and rules

  - Team inspects and adapts (continuous improvement) the overall process
  - ✗ Team Coach/Scrum Master exists, is full-time, and is effective
  - The team has an effective channel for obstacle escalation

- Testing Practices
  - All testing is done within the iteration and does not lag behind
  - Iteration defects are fixed within that iteration
  - Unit tests written before development
  - Acceptance tests written before development
  - 100% automated unit test coverage
  - Automated acceptance tests

- Development Practices/Infrastructure
  - Source control system exists
  - Continuous build with 100% successful builds
  - Developers integrate code multiple times per day
  - Team has administrative access to their own workstations
  - Team has administrative control over their development environment
  - Team is permitted to refactor anywhere in the code base
  - Adequate and effective code review practices
  - Coding standards exist and applied
  - Stories accepted and demonstrated on integrated build
  - Refactoring is continuous
  - Pair programming is practiced
  - Identical builds for developers' workstations

D

Mapping

D.1 Questions to Practices/Strategies

In order to distinguish the questions among the tools the following annotation has been used.

★ Team Agility Assessment - Direct match

☆ Team Agility Assessment - Relevant match

❄ Team Agility Assessment - Non Relevant

✱ Perceptive Agile Management - Direct match

✳ Perceptive Agile Management - Relevant match

Iterative and Incremental Development

★ Stories sufficiently elaborated prior to planning meetings

☆ Team discusses acceptance criteria during iteration planning

✱ We kept the iteration deadlines

✳ The software delivered at iteration end always met quality requirements of production code

✳ The team rather reduced the scope than delayed the deadline

✳ At the end of an iteration, we delivered a potentially shippable product

✳ Working software was the primary measure for project progress

✯ To what extent are the estimates for the amount of work to be done during each iteration accurate?

✯ To what extent are the stories fully estimated when added to the iteration backlog?

✯ To what extent are the stories prioritized when added to the iteration backlog?

Product Backlog

★ Backlog prioritized and ranked by business value

☆ Backlog estimated at gross level

✯ To what extent is a product backlog maintained?

✯ To what extent are the features prioritized when they are added to the product backlog?

✯ To what extent are the features fully estimated before they are added to the product backlog?

Smaller and Frequent Product Releases

★ The team has small and frequent releases

✱ We implemented our code in short iterations

✯ 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 iteration is four weeks or less)

✯ To what extent is the product developed so far in-sync with the customers' needs and expectations?

✯ To what extent are the deployments not rolled back?

Customer/User Acceptance Testing

✱ The customer provided a comprehensive set of test criteria for customer acceptance

✳ How often did you apply customer acceptance tests?

✳ The customer focused primarily on customer acceptance tests to determine what had been accomplished at the end of an iteration

✳ Customer acceptance tests were used as the ultimate way to verify system functionality and customer requirements

✳ A requirement was not regarded as finished until its acceptance tests (with the customer) had passed

Constant Velocity

☆ Team works at a sustainable pace

Iteration Progress Tracking and Reporting

★ Iteration backlog defined

★ Iteration backlog ranked by priority

★ Iteration planning meeting attended and effective

★ Iterations are no more than four weeks in length

★ Iterations are of a consistent fixed length

☆ Iteration theme established and communicated

☆ Team velocity measured and used for planning

☆ Release theme established and communicated

☆ Release planning meeting attended and effective

☆ Release backlog defined

☆ Release backlog ranked by priority

☆ Release backlog estimated at plan level

☆ Release progress tracked by feature acceptance

☆ The whole team is present at release planning meetings

☆ Iteration progress tracked by task to do (burn-down chart) and card acceptance (velocity)

✳ All members of the technical team actively participated during iteration planning meetings

✳ All technical team members took part in defining the effort estimates for requirements of the current iteration

✳ All concerns from team members about reaching the iteration goals were considered

✳ When effort estimates differed, the technical team members discussed their underlying assumption

- ✳ The effort estimates for the iteration scope items were modified only by the technical team members
- ✯ To what extent is an iteration backlog maintained?
- ✯ To what extent is the length of an iteration 4 weeks or less?
- ✯ To what extent are the iterations timeboxed?
- ✯ To what extent are the release cycles timeboxed?
- ✯ To what extent are only a subset of the identified features developed during a release cycle?

Self-Organizing Teams

- ★ Team leads communication; communication not managed
- ★ Team defines, estimates, and selects their own work (stories and tasks)
- ☆ Team develops and manages iteration backlog
- ☆ Team manages interdependencies and constraints
- ☆ Team has administrative access to their own workstations
- ☆ Team has administrative control over their development environment
- ☆ Team is 100% dedicated to the release (no time-slicing)
- ☆ Team meets its commitments to release
- ☆ The team has a common language and metaphor to describe the release
- ☆ Team self-polices and reinforces use of agile practices and rules
- ☆ Team is smaller than 15 people
- ✱ Each developer signed up for tasks on a completely voluntary basis
- ✯ 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?
- ✯ 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?

✯ To what extent does the management support the self-managing nature of the teams?

Appropriate Distribution of Expertise

★ Team is cross-functional with integrated product owner, development, documentation and QA

★ Team members complete commitments

✯ 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?

High-Bandwidth Communication

★ Team is collocated

★ Team works in a physical environment that fosters collaboration

★ Release planning meeting attended and effective

☆ The team has an effective channel for obstacle escalation

✱ The customer was reachable

✱ The developers could contact the customer directly or through a customer contact person without any bureaucratical hurdles

✱ Developers were located majorly in ...

✱ All members of the technical team (including QA engineers, db admins) were located in ...

✱ Requirements engineers were located with developers in ...

✱ The project/release manager worked with the developers in ...

✱ The customer was located with the developers in ...

✲ The developers had responses from the customer in a timely manner

✯ To what extent is the product developed so far in-sync with the customers' needs and expectations?

✯ 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 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?

✯ To what extent does open communication prevail between the business and the development team?

✯ To what extent does open communication prevail between the manager 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 development team?

✯ To what extent does open communication prevail between members of different teams?

Daily Progress Tracking Meetings

☆ Daily stand up on time, fully attended and effectively communicates

✱ Stand up meetings were to the point, focusing only on what had been done and needed to be done on that day

✲ Stand up meetings were extremely short (max. 15 minutes)

- ⁜ All relevant technical issues or organizational impediments came up in the stand up meetings
- ⁜ Stand up meetings provided the quickest way to notify other team members about problems
- ⁜ When people reported problems in the stand up meetings, team members offered to help instantly
- ✯ To what extent is the time allocated for the daily progress tracking meetings utilized effectively?

Retrospective Meetings

- ★ Iteration review meeting attended and effective
- ★ Release review meeting attended and effective
- ☆ Team inspects and adapts (continuous improvement) the overall process
- ☆ Team inspects and adapts (continuous improvement) the Iteration Plan
- ☆ Team inspects and adapts (continuous improvement) the release plan
- ⁜ How often did you apply retrospectives?
- ⁜ The retrospectives helped us become aware of what we did well in the past iteration(s)
- ⁜ The retrospectives helped us become aware of what we should improve in the upcoming iteration(s)
- ⁜ All team members actively participated in gathering lessons learned in the retrospectives
- ⁜ In the retrospectives (or shortly afterwards), we systematically assigned all important points for improvement to responsible individuals
- ⁜ Our team followed up intensively on the progress of each improvement point elaborated in a retrospective
- ✯ 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?
- ✯ To what extent were the retrospective goals set during the previous iteration met?

Test Driven Development

★ Unit tests written before development

★ 100% automated unit test coverage

☆ Acceptance tests written before development

✱ New code was written with unit tests covering its main functionality

✱ The implemented code was written to pass the test case

✳ Automated unit tests sufficiently covered all critical parts of the production code

✯ To what extent did the developers provide adequate code coverage from the tests?

✯ To what extent is the product developed so far in-sync with the customers' needs and expectations?

✯ To what extent do developers write tests first before writing code?

✯ To what extent are the test plans created before the developers start coding?

Refactoring

★ Refactoring is continuous

☆ Team is permitted to refactor anywhere in the code base

✳ There were enough unit tests and automated system tests to allow developers to safely change any code

✯ 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?

✯ To what extent does the management support the implementation of refactoring?

Software Configuration Management

★ Source control system exists

✯ To what extent do teams use appropriate tools for version control and management?

Adherence to Standards

★ Coding standards exist and applied

★ Pair programming is practised

★ Adequate and effective code review practices

✯ 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

★ Automated acceptance tests

☆ Developers integrate code multiple times per day

☆ Stories accepted and demonstrated on integrated build

☆ Continuous build with 100% successful builds

☆ Identical builds for developers' workstations

✱ Developers had the most recent version of code available

✱ Code was checked in quickly to avoid code synchronization/integration hassles

✳ The team integrated continuously

✳ All unit tests were run and passed when a task was finished and before checking in and integrating

✳ For detecting bugs, test reports from automated unit tests were systematically used to capture the bugs

✯ 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 are automated test suites developed?

✯ To what extent are the code bases not shared?

✯ To what extent do automated builds run one or more times everyday?

✯ To what extent has each story been accepted by the customer?

Client-Driven Iterations

✱ The customer picked the priority of the requirements in the iteration plan

✳ When the scope could not be implemented due to constraints, the team held active discussions on re-prioritization with the customer on what to finish within the iteration

✯ To what extent do the customers establish the priorities of the story?

✯ To what extent is the product developed so far in-sync with the customers' needs and expectations?

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

Minimal or Just Enough 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?

Continuous Feedback

✳ The feedback from the customer was clear and clarified his requirements or open issues to the developers

✯ To what extent do the customers provide feedback to the business and the development team?

✯ To what extent is the product developed so far in-sync with the customers' needs and expectations?

Evolutionary Requirements

✯ To what extent are the features reprioritized as and when new features are identified?

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

✯ To what extent are only the high level features identified upfront?

✯ To what extent are the architecture requirements allowed to evolve over time?

D.2 Non Relevant Questions

- ❄ Product owner defines acceptance criteria for stories
- ❄ Product owner and stakeholders participate at iteration and release planning
- ❄ Product owner and stakeholders participate at iteration and release review
- ❄ Product owner collaboration with team is continuous
- ❄ Team completes and product owner accepts the release by the release date
- ❄ Work is not added by the product owner during the iteration
- ❄ Team completes and product owner accepts the iteration
- ❄ All testing is done within the iteration and does not lag behind
- ❄ Iteration defects are fixed within that iteration
- ❄ Team Coach/Scrum Master exists, is full-time, and is effective

E

OPS, PAM, TAA

E.1 Capability

- Refactoring
  - Support for Refactoring
    - Is refactoring an expected activity?
    - Is it feasible to implement code refactoring?
    - Is it feasible to implement architecture refactoring?
  - Buy-in for Refactoring
    - Are the teams receptive to implementing refactoring?
    - ★ Are the teams permitted to refactor anywhere in the code base?
    - 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?
    - ★ Are the developers expected to write acceptance tests first for their code?

  - 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 bugs/enhancements, 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 bugs/enhancements 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 implementing 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?
    - ★ Size
        - ★ Is it expected the team be smaller than 15 people?
    - ★ Communication
        - ★ Are the team members expected to have a common language?
    - ★ System Administration
        - ★ Are the team members expected to have administrative access to their own workstations?
        - ★ Are the team members expected to have control over their development environment?

- 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 time-frames
    - 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?
    - ★ Is it expected that the team velocity is used for planning?
  - Estimation Authority
    - Are the developers expected to estimate the time required to complete each story?
  - 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?
  - Requirements Management
    - Are tools available for managing the bugs/enhancements?
  - 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?

★ Velocity

  ★ Progress Estimation

★ Is it expected that the progress is track by a burn down chart and by measuring velocity?

- Evolutionary Requirements
  - Minimal Big Requirements Up Front and Big Design Up Front
    - 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?
  - Just-In-Time 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 release cycle into stories be defined?
- Minimal Documentation
  - Tool Support for Minimal Documentation
    - Do tools for maintaining documentation exist?
  - Process 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?

E.2 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?

  - ★ To what extent are the teams permitted to refactor anywhere in the code base?
  - ★ To what extent were there enough unit tests and automaated system test to allow developers to safely refactor?

- 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?
    - ★ To what extent did the unit tests cover all critical parts of the production code?

- 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?
  - ★ Team
    - ★ To what extent did team members participate in the retrospective meetings?
    - ★ To what extent does the team inspect and adapt the overall process?
    - ★ To what extent does the team continuously improve the iteration plan?
    - ★ To what extent does the team continuously improve the release plan?

- ★ Outcomes
  - ★ To what extent were the retrospectives helpful for seeing what worked well in the past iterations?
  - ★ To what extent were the retrospectives helpful for seeing what should be improved in the upcoming iterations?
  - ★ To what extent did the points improve from the previous retrospectives?
  - ★ To what extent were important points for improvement assigned to responsible individuals?

- 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?
    - ★ To what extent is the team cross-functional?

- 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?
    - ★ Process Support for Continuous Integration
        - ★ To what extent do the team members integrate their code per day?
  - 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 unit tested successfully?
    - 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?
    - ★ To what extent were the stories accepted and demonstrated on integrated build?
  - Daily/Frequent builds
    - To what extent do automated builds run one or more times everyday?
    - ★ To what extend are the automated builds successful?
    - ★ To what extend test reports for automated unit tests were systematically used to capture the bugs?

- 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?
    - ★ To what extend does the team self-police and reinforce the practices and rules?

  - Management support
    - To what extent does the management support the self-managing nature of the teams?
  - ★ Commitment
    - ★ To what extent is the team dedicated to the release?

- 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?
  - ★ Customer Acceptance
    - ★ To what extent were customer acceptance tests applied?
    - ★ To what extent did the customer focus on acceptance tests to determine what had been accomplished?
    - ★ To what extent were the acceptance tests the ultimate way to verify system functionality and customer requirements?

- High-bandwidth communication
  - ★ On-site Customer
    - ★ To what extent were the responses from the customer in a timely manner?
    - ★ To what extent was the feedback from the customer clear and clarified the requirements or open issues to the developers?
  - 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 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?

  ★ Daily progress tracking meetings

    - To what extent is the time allocated for the daily progress tracking meetings utilized effectively?
    ★ To what extent were daily progress tracking meetings up to 15 minutes?
    ★ To what extent did all relevant technical issues or organizational impediments came up in the daily progress tracking meetings?
    ★ To what extent were the daily progress tracking meetings the quickest way to notify the other team members about problems?
    ★ To what extent did team members offer to help when people reports problems in the daily progress tracking meetings?
    ★ To what extent was the daily progress tracking meeting on time?

  – 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 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 development team?
    - To what extent does open communication prevail between members of different teams?
    ★ To what extent does the team have an effective channel for obstacle escalation?

- Client-driven Iterations

  – Requirements Prioritization

    - To what extent do the customers establish the priorities of the story?
    ★ To what extent are the features reprioritized when the scope could not be implemented due to constraints?

  – 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 time-frames
    - 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 iteration 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?
    - ★ To what extent did the team reduce the scope than delaying the deadline?
    - ★ To what extent did team members take part in defining the amount of work to be done?
    - ★ To what extent did the concerns of the team members about reaching the iteration goals were taken into consideration?
  - 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 list maintained?
    - ★ To what extent do the team members develop and manage an iteration backlog?
    - To what extent are the stories fully estimated when added to the list?
    - To what extent are the stories prioritized when added to the list?
  - ★ Planning
    - ★ To what extent did the team discuss the acceptance criteria during the iteration planning?
    - ★ To what extent did the team members actively participate during iteration planning meetings?

- ★ Product
  - ★ To what extent was the product shippable at the end of the iteration?
  - ★ To what extent was the working software the primary measure for project progress?
  - ★ To what extent did the delivered software meet the quality requirements of production code at the end of the iteration?

- 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 product backlog?
    - To what extent are the features fully estimated before they are added to the product backlog?
    - ★ To what extent is a release backlog maintained?
    - ★ To what extent are the features prioritized when they are added to the release 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?

★ Velocity

- ★ Pace
  - ★ To what extent does the team work at a sustainable pace?

- 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?
    - ★ To what extent is a requirement regarded as finished until its acceptance tests (with the customer) has passed?
  - Minimal Big Requirements Up Front and Big Design Up Front
    - 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?

F

Data Plots

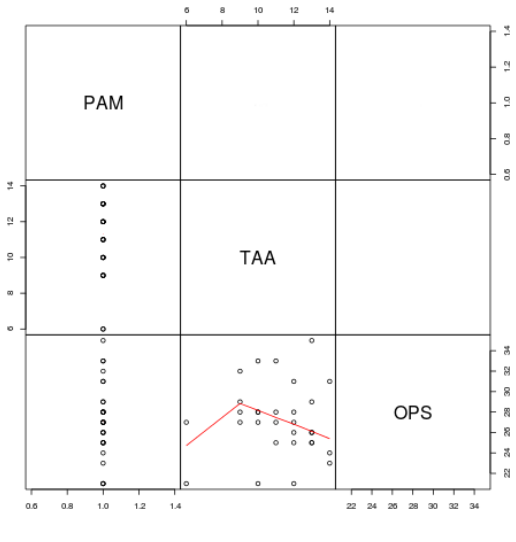

Figure F.1: Appropriate Distribution of Expertise

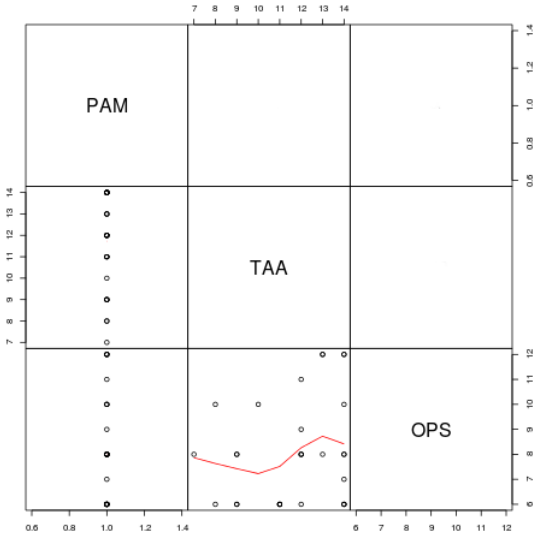

Figure F.2: Adherence to Standards

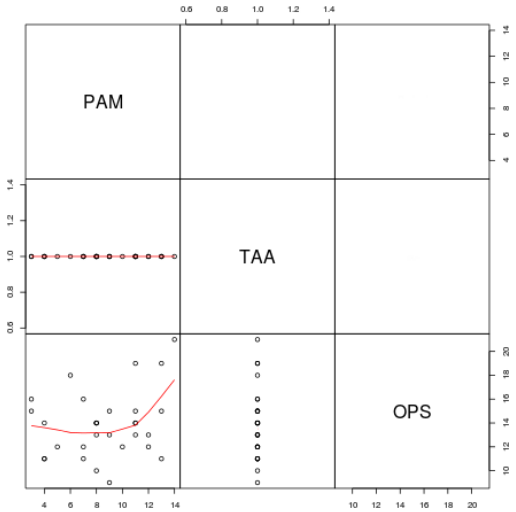

Figure F.3: Client-Driven Iterations

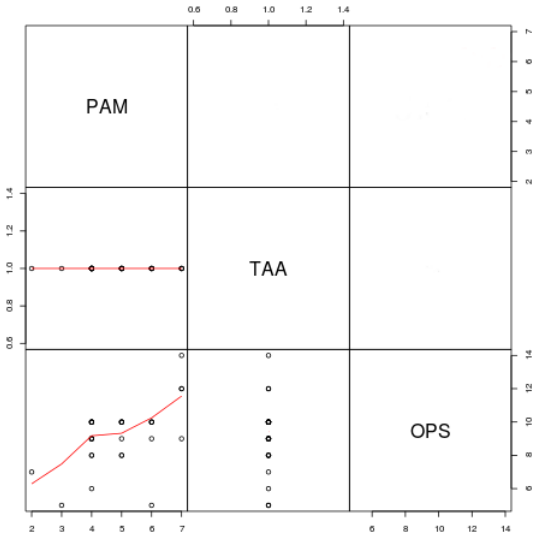

Figure F.4: Continuous Feedback

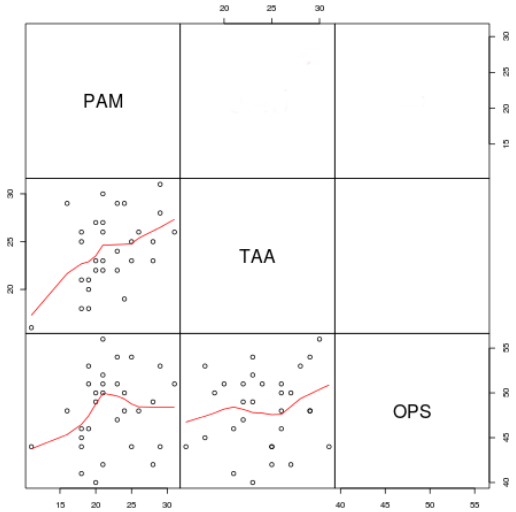

Figure F.5: Continuous Integration

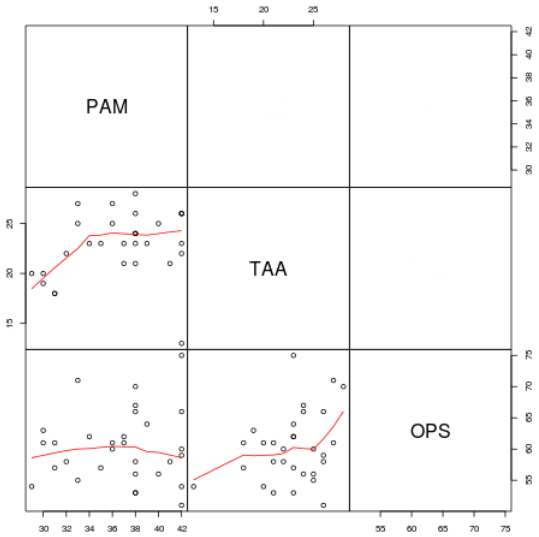

Figure F.6: High-Bandwidth Communication

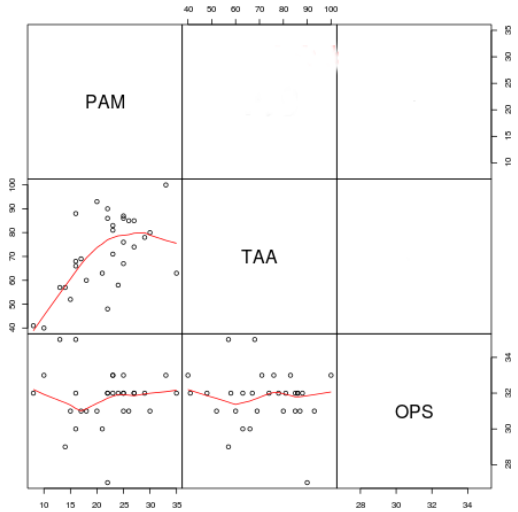

Figure F.7: Iteration Progress Tracking and Reporting

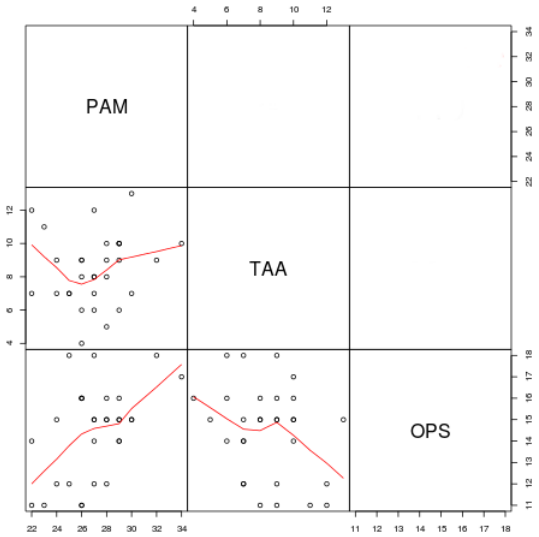

Figure F.8: Iterative and Incremental Development

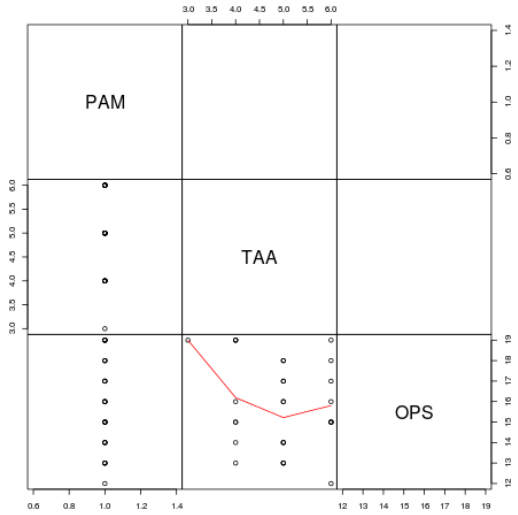

Figure F.9: Product Backlog

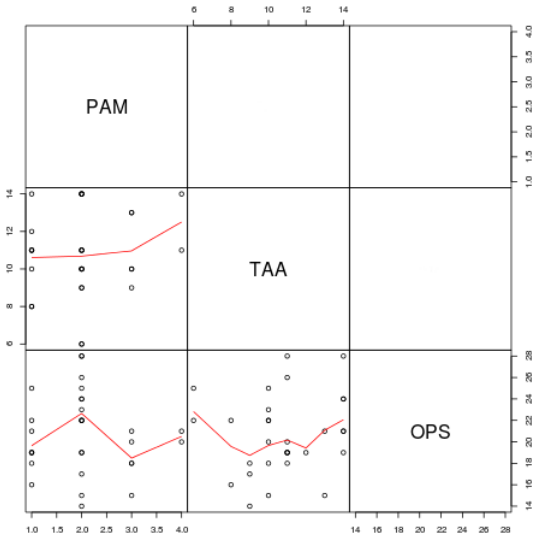

Figure F.10: Refactoring

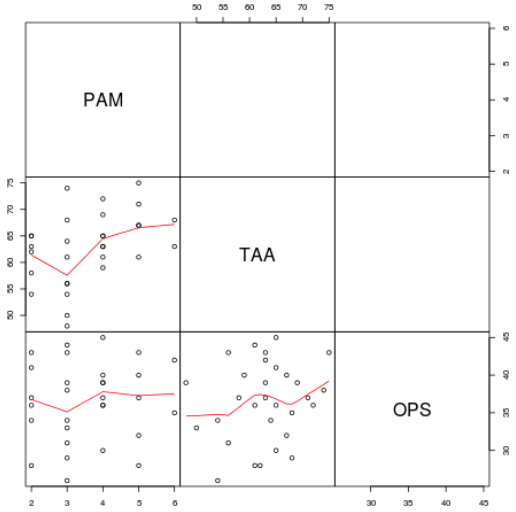

Figure F.11: Self-Organizing Teams

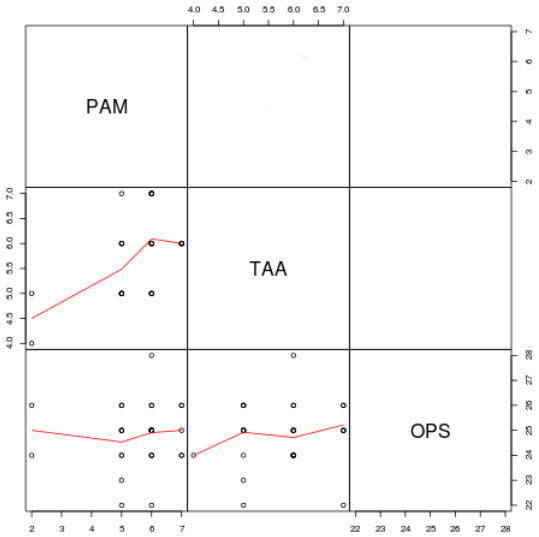

Figure F.12: Smaller and Frequent Product Releases

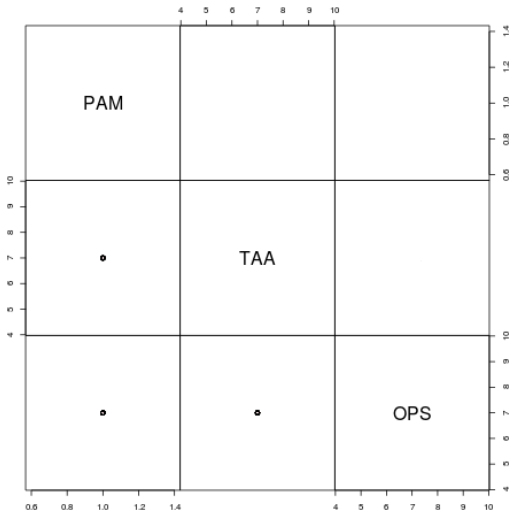

Figure F.13: Software Configuration Management

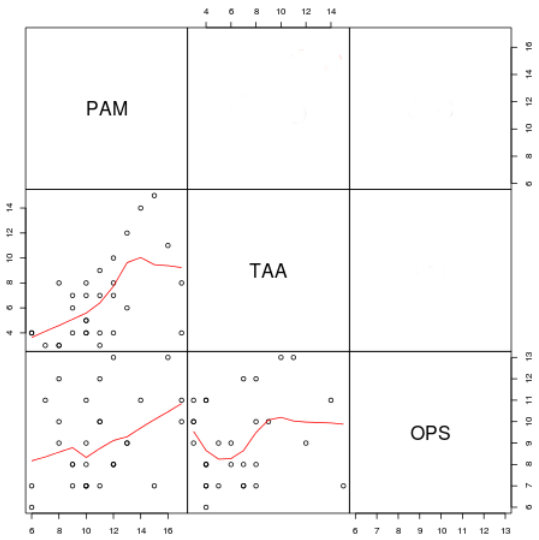

Figure F.14: Test Driven Development

Bibliography

[1] Ambler, S. W., 2011. "Has agile peaked?".

[2] Ambysoft, 2013. "How agile are you? 2013 survey".
URL http://www.ambysoft.com/surveys/howAgileAreYou2013.html#Figure6

[3] Arthur, J., Nance, R., December 1990 1990. A framework for assessing the adequacy and effectiveness of software development methodologies. In: Proceedings of the Fifteenth Annual Software Engineering Workshop. Greenbelt, MD.

[4] Aveling, B., 2004. Xp lite considered harmful? In: Eckstein, J., Baumeister, H. (Eds.), Extreme Programming and Agile Processes in Software Engineering. Vol. 3092. Springer Berlin Heidelberg, pp. 94–103.

[5] Beck, K., Andres, C., 2004. Extreme Programming Explained: Embrace Change (2Nd Edition). Addison-Wesley Professional.

[6] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mallor, S., Shwaber, K., Sutherland, J., 2001.
URL http://agilemanifesto.org/principles.html

[7] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2001. Manifesto for agile software development.
URL http://www.agilemanifesto.org/

[8] Boehm, B., Turner, R., June 2003. Observations on balancing discipline and agility. In: Agile Development Conference, 2003. ADC 2003. Proceedings of the. pp. 32–39.

[9] Boehm, B., Turner, R., May 2004. Balancing agility and discipline: evaluating and integrating agile and plan-driven methods. In: Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on. pp. 718–719.

[10] Cockburn, A., 2002. Agile software development. Agile software development series. Addison-Wesley.
URL `http://books.google.se/books?id=JxYQ1Zb61zkC`

[11] Cockburn, A., 2004. Crystal Clear a Human-powered Methodology for Small Teams, 1st Edition. Addison-Wesley Professional.

[12] Comparative-Agility, 2008.
URL `http://comparativeagility.com`

[13] Conboy, K., Fitzgerald, B., 2004. Toward a conceptual framework of agile methods: A study of agility in different disciplines. In: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research. WISER '04. pp. 37–44.

[14] Datta, S., 2009. Metrics and techniques to guide software development. Ph.D. thesis, Florida State University College of Arts and Sciences.

[15] Delestras, S., Roustit, M., Bedouch, P., Minoves, M., Dobremez, V., Mazet, R., Lehmann, A., Baudrant, M., Allenet, B., feb 2013. Comparison between two generic questionnaires to assess satisfaction with medication in chronic diseases. PLoS ONE 8 (2), e56247.

[16] Drive, G., 2012.

[17] Escobar-Sarmiento, V., Linares-Vasquez, M., Oct 2012. A model for measuring agility in small and medium software development enterprises. In: Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En. pp. 1–10.

[18] Fowler, M., 2006.
URL `http://martinfowler.com/`

[19] Gren, L., 2013. Is the group ready to go agile? Ph.D. thesis, Chalmers University of Technology.

[20] Highsmith, III, J. A., 2000. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Publishing Co., Inc., New York, NY, USA.

[21] Hossain, E., Ali Babar, M., Verner, J., 2009. Towards a framework for using agile approaches in global software development. In: Product-Focused Software Process Improvement. Vol. 32. Springer Berlin Heidelberg, pp. 126–140.

[22] IBM, 2010.
URL `http://www-01.ibm.com/support/docview.wss?uid=swg21482329`

[23] Ikoma, M., Ooshima, M., Tanida, T., Oba, M., Sakai, S., May 2009. Using a validation model to measure the agility of software development in a large software development organization. In: Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. pp. 91–100.

[24] Jalali, S., 2012. Efficient software development through agile methods. Ph.D. thesis, Blekinge Institute of Technology.

[25] Jalali, S., Wohlin, C., Angelis, L., 2014. Investigating the applicability of agility assessment surveys: A case study. Journal of Systems and Software (0), –.

[26] Jalali, S., Wohlin, C., Angelis, L., 2014. Investigating the applicability of agility assessment surveys: A case study. Journal of Systems and Software 98 (0), 172 – 190.

[27] Kara, S., Kayis, B., 2004. Manufacturing flexibility and variability: an overview. Journal of Manufacturing Technology Management 15 (6), 466–478.

[28] Kidd, P. T., 1994. Agile Manufacturing: Forging New Frontiers.

[29] Koch, A., 2005. Agile Software Development: Evaluating The Methods For Your Organization. Artech House computing library. Artech House, Incorporated.
URL `http://books.google.se/books?id=vJ99QgAACAAJ`

[30] Lappo, P., Andrew, H. C. T., 2004. Assessing agility. Vol. 3092 of Lecture Notes in Computer Science. Springer, pp. 331–338.

[31] Leffingwell, D., 2007. Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series). Addison-Wesley Professional.

[32] Livermore, J., March 2006. What elements of xp are being adopted by industry practitioners? In: SoutheastCon, 2006. Proceedings of the IEEE. pp. 149–152.

[33] Nagel, R. N., Dove, R., 1991. 21st Century Manufacturing Enterprise Strategy: An Industry-Led View. Diane Pub Co.

[34] Nagel, R. N., Dove, R., 1991. 21st Century Manufacturing Enterprise Strategy: An Industry-Led View.

[35] Nietzsche, F., 2012. Thus Spoke Zarathustra. Simon & Brown.

[36] Palmer, S. R., Felsing, M., 2001. A Practical Guide to Feature-Driven Development, 1st Edition. Pearson Education.

[37] Patel, C., Lycett, M., Macredie, R., de Cesare, S., Jan 2006. Perceptions of agility and collaboration in software development practice. In: System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on. Vol. 1. pp. 10c–10c.

[38] Poonacha, K., Bhattacharya, S., Jan 2012. Towards a framework for assessing agility. In: System Science (HICSS), 2012 45th Hawaii International Conference on. pp. 5329–5338.

[39] Qumer, A., Henderson-Sellers, B., 2006. Measuring agility and adaptibility of agile methods: A 4 dimensional analytical tool.

[40] Qumer, A., Henderson-Sellers, B., Nov. 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. J. Syst. Softw. 81 (11), 1899–1919.

[41] Ramesh, G., Devadasan, S., 2007. Literature review on the agile manufacturing criteria. Journal of Manufacturing Technology Management 18 (2), 182–201.

[42] Razali, N., Wah, Y. B., Jun. 2011. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. Journal of Statistical Modeling and Analytics 2 (1).

[43] Reifer, D. J., 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. Springer-Verlag, pp. 185–196.

[44] RStudio, 2010.
URL `http://www.rstudio.com`

[45] Sahota, M., 2012. An Agile Adoption And Transformation Survival Guide. lulu.com.

[46] Schwaber, K., Beedle, M., 2001. Agile Software Development with Scrum (Series in Agile Software Development). Prentice Hall.

[47] Shawky, D., Ali, A., Nov 2010. A practical measure for the agility of software development processes. In: Computer Technology and Development (ICCTD), 2010 2nd International Conference on. pp. 230–234.

[48] Sidky, A., 2007. A structured approach to adopting agile practices: The agile adoption framework. Ph.D. thesis, Virginia Polytechnic Institute and State University.

[49] Sidky, A., Arthur, J., Bohner, S., 2007. A disciplined approach to adopting agile practices: the agile adoption framework. Innovations in systems and software engineering 3 (3), 203–216.

[50] So, C., Scholl, W., 2009. Perceptive agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of agile practices. Vol. 31 of Lecture Notes in Business Information Processing. Springer, pp. 83–93.

[51] Soundararajan, S., 2013. Assessing agile methods, investigating adequacy, capability and effectiveness. Ph.D. thesis, Virginia Polytechnic Institute and State University.

[52] Soundararajan, S., Arthur, J., Balci, O., Aug 2012. A methodology for assessing agile software development methods. In: Agile Conference (AGILE), 2012. pp. 51–54.

[53] Sutherland, J., 2008. The scrum but test.
URL `http://antoine.vernois.net/scrumbut/?page=intro&lang=en`

[54] Taromirad, M., Ramsin, R., Oct 2008. Cefam: Comprehensive evaluation framework for agile methodologies. In: Software Engineering Workshop, 2008. SEW '08. 32nd Annual IEEE. pp. 195–204.

[55] Taylor, P., Greer, D., Sage, P., Coleman, G., McDaid, K., Lawthers, I., Corr, R., 2006. Applying an agility/discipline assessment for a small software organisation. In: Product-Focused Software Process Improvement. Vol. 4034. pp. 290–304.

[56] ThoughtWorks-Studio, 1993.
URL `http://www.agileassessments.com`

[57] Tsourveloudis, N., Valavanis, K., 2002. On the measurement of enterprise agility. Journal of Intelligent and Robotic Systems 33 (3), 329–342.

[58] VersionOne, 2013. 8th annual state of agile survey.
URL `http://stateofagile.versionone.com/`

[59] Waters, K., 2008. How agile are you?
URL `http://www.allaboutagile.com/how-agile-are-you-take-this-42-point-test/`

[60] Williams, L., Krebs, W., Layman, L., Antón, A. I., Abrahamsson, P., 2004. Toward a framework for evaluating extreme programming.

[61] Williams, L., Rubin, K., Cohn, M., Aug 2010. Driving process improvement via comparative agility assessment. In: Agile Conference (AGILE), 2010. pp. 3–10.

[62] Yauch, C. A., 2011. Measuring agility as a performance outcome. Journal of Manufacturing Technology Management 22 (3), 384–404.