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

Acknowledgements

Contents

1

Introduction

2

Related Work

2.1 Introduction

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 many papers of this field.

2.2 Balancing Discipline and Agility

Boehm and Turner [4] did not come up with a tool to measure agility but rather to balance between agility and discipline. Accoring to them [5] discipline is the foundation for any successful endeavor and creates experience, history and well-organized memories. On the other hand agility is described as a counterpart of descipline. 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 [4] in their research found that there are 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 which are the:

- size of a team working in a project
- criticallity of damage of unexpected defects
- culture on how to balance between chaos an order
- dynamism
- personnel which refers to the extended Cockburn [6] skill rating

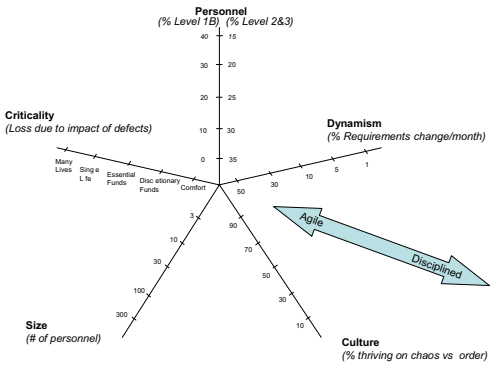

Figure 2.1: Dimensions affecting method selection

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented newsituation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Table 2.1: Levels of software method understanding and use (after Cockburn)

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

2.3 4-Dimensional Analytical Tool

Qumer and Henderson-Sellers [14] created the 4-Dimensional Analytical Tool (4-DAT) for analysing and comparing agile methods. The objective of the tool is to provide a mechanism to assess the degree of agility and adoptability of any agile methodology. The measurements are taken at a specific level in a process and using specific practices.

2.3.1 Dimension 1 - Method Scope Characterization

The first dimension describes the key scope items which have been derived from their literature review based on Beck and Andres [2], Koch [10], Palmer and Felsing [12], Highsmith [9] and provides a method comparison at a high level.

These items are: a) Project Size b) Team Size c) Development Style d) Code Style e) Technology Environment f) Physical Environment g) Business Culture h) Abstraction Mechanism

The aforementioned elements are considered essential for supporting the method used by a team or organisation. Table 2.2 provides a description for the items.

Scope	Description
1. Project Size	Does the method specify support for small, medium or large projects (business or other)?
2. Team Size	Does the method support for small or large teams (single or multiple teams)?
3. Development Style	Which development style (iterative, rapid) does the method cover?
4. Code Style	Does the method specify code style (simple or complex)?
5. Technology Environment	Which technology environment (tools, compilers) does the method specify?
6. Physical Environment	Which physical environment (co-located or distributed) does the method specify?
7. Business Culture	What type of business culture (collaborative, cooperative or non-collaborative) does the method specifiy?
8. Abstraction Mechanism	Does the method specify abstraction mechanism (object-oriented, agent-oriented)?

Table 2.2: 4-DAT Dimension 1

2.3.2 Dimension 2 - Agility Characterization

The second dimesion is the only quantitavie 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 the following five variables. Table 2.3 provides a description for them. a) Flexibility b) Speed c) Leanness d) Learning e) Responsiveness

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, otherwise 0. Qumer and Henderson-Sellers [14] define the degree of agility (DA) as "the fraction of the five agility variables that are encompassed and supported".

The function for calculating the DA is the following
$DA\ (Object) = (1/m) \sum m\ DA(Object,\ Phase\ or\ Practices)$

Features	Description
1. Flexibility	Does the method accommodate expected or unexpected changes?
2. Speed	Does the method produce results quickly?
3. Leanness	Does the method follow shortest time span, use economical, simple and quality instruments for production?
4. Learning	Does the method apply updated prior knowledge and experience to learn?
5. Responsiveness	Does the method exhibit sensitiveness?

Table 2.3: 4-DAT Dimension 2

2.3.3 Dimension 3 - Agile Values Characterization

The third dimension consists of six agile values. Four of them are derived directly from the Agile Manifesto [3], while the fifth comes from [10]. The last value is suggested by Qumer and Henderson-Sellers [14] after having studied several agile methods. Table 2.5 shows the agile values.

Agile values	Description
1. Individuals and interactions over processes and tools	Which practices value people and interaction over processes and tools?
2. Working software over comprehensive documentation	Which practices value working software over comprehensive documentation?
3. Customer collaboration over contract negotiation	Which practices value customer collaboration over contract negotiation?
4. Responding to change over following a plan	Which practices value responding to change over following a plan?
5. Keeping the process agile	Which practices helps in keeping the process agile?
6. Keeping the process cost effective	Which practices helps in keeping the process cost effective?

Table 2.4: 4-DAT Dimension 3

2.3.4 Dimension 4 - Software Process Characterization

The fourth dimension examines the practices that support four processes as these are presented by Qumer and Henderson-Sellers [14]. Table 2.5 lists these processess.

Process	Description
1. Development Process	Which practices cover the main life cycle process and testing (Quality Assurance)?
2. Project Management Process	Which practices cover the overall management of the project?
3. Software Configuration Control Process / Support Process	Which practices cover the process that enables configuration management?
4. Process Management Process	Which practices cover the process that is required to manage the process itself?

Table 2.5: 4-DAT Dimension 4

2.4 Escobar

Escobar-Sarmiento and Linares-Vasquez [8] 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, except from the last one.

- Agile Project Management Assessment - proposed by Qumer and Henderson-Sellers [14]
- Project Agility Assessment - proposed by Taylor et al. [16]
- Workteam Agility Assessment - proposed by Leffingwell [11]
- 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.

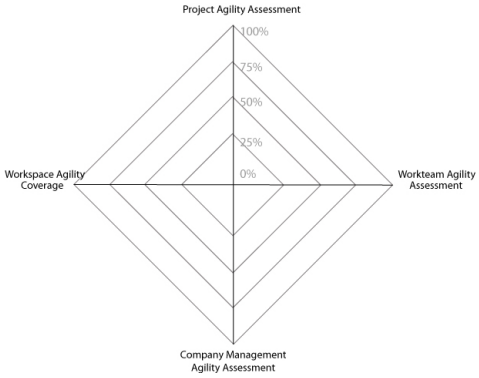

Figure 2.2: ??????????

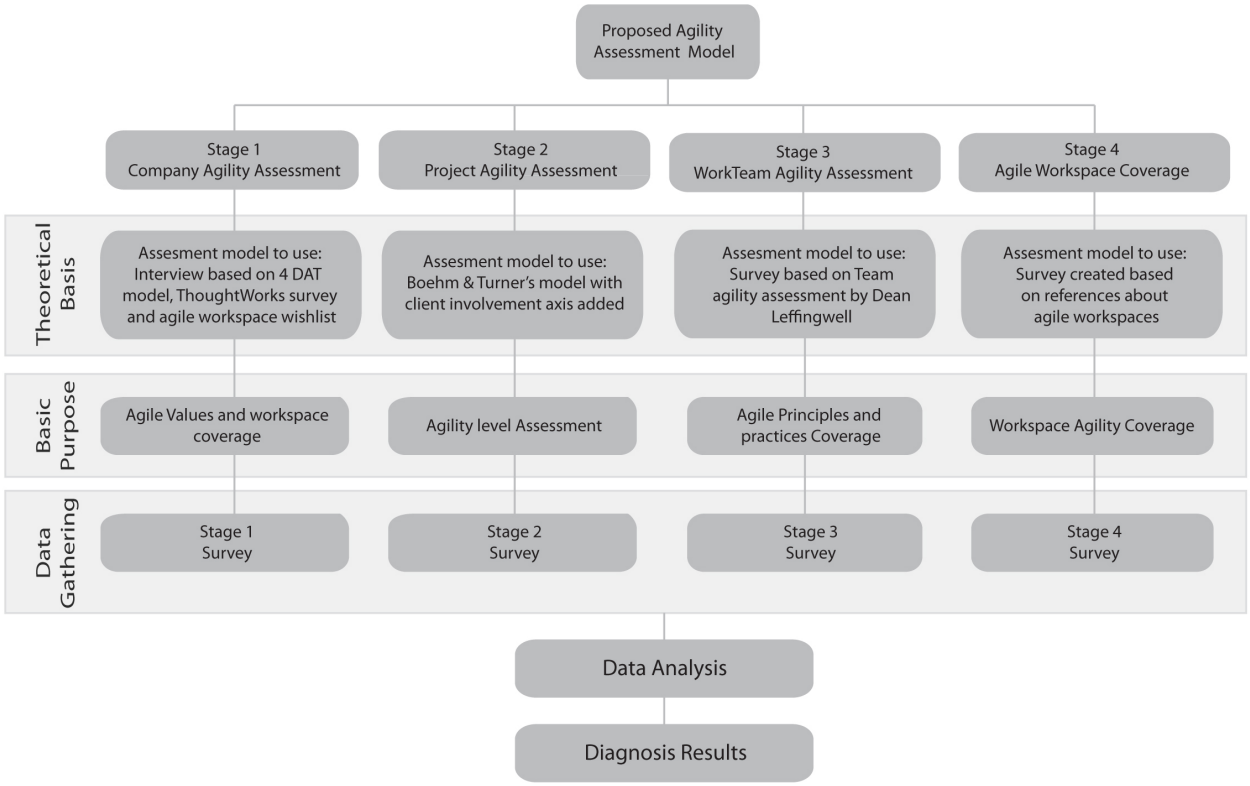

Figure 2.3: ??????????

2.5 Sidky

2.6 OPS Framework

Soundararajan [15] created the Objectives, Principles and Stategies (OPS) Framework in order to assess the "goodness" of an agile methodology. The focus of this tool is mainly on eXtreme Programming [2], Feature Driven Development (FDD) [12], Lean [13], Crystal [7] 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 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

The OPS Framework identifies

- Objectives of the agile philosophy - "something aimed at or striven for" as defined by [1]
- Principles - what rules a process in order to achieve an objective according to [1]
- Strategies - the implementations of the principles (i.e. they are the means for achieving the principles)
- Linkages - the connectors between a) the objectives and principles, b) the principles and the strategies. The linkages show the path in order to asses the adequacy, capability and effectiveness of the method used.
- Indicators for assessing the extent to which an organization supports the implementation and effectiveness of those strategies - In order to measure the capability and the effectiveness the strategies use properties which contain a number of questions. These properties differ for the capability and the effectiveness. Indicator is named the combination of a strategy with a property. They are directly measurable and are tailored to assess the strategies

The OPS Framework identifies in total 5 objectives, 9 principles, 17 strategies 54 linkages and 80 indicators.

2.7 Thoughtworks

Thoughtworks [**?**] is a worldwide consulting company. They have developed an online survey for assessing agility. People can answer to the survey and they will get a report evaluating at which level their team or company is.

2.8 Other

Taylor et al. [16] modified the tool created by Boehm and Turner [4] by adding a sixths axis for the *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.

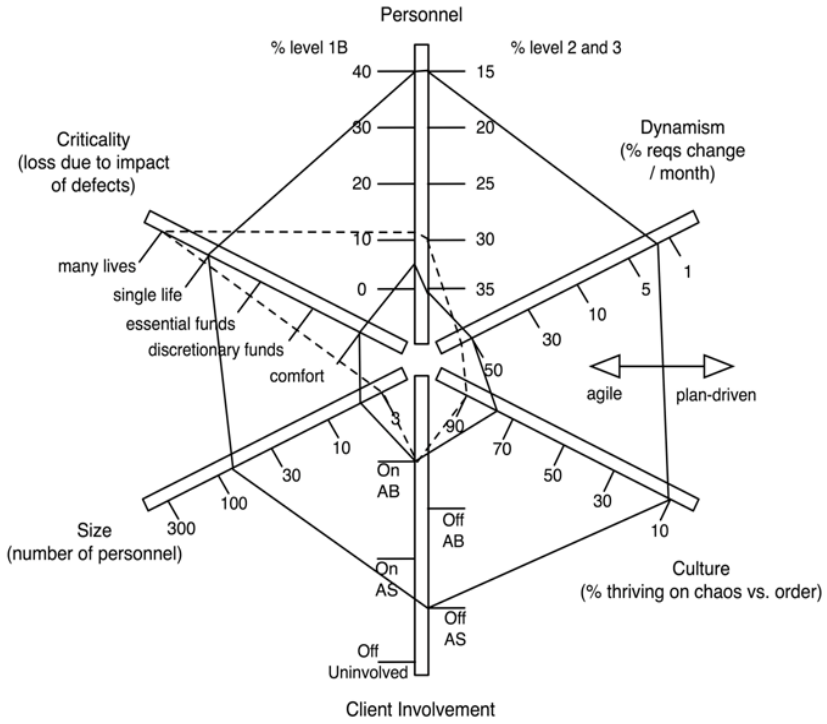
Personnel
% level 1B
% level 2 and 3
40
30
20
10
0
15
20
25
30
35
Criticality
(loss due to impact
of defects)
many lives
single life
essential funds
discretionary funds
comfort
Dynamism
(% reqs change
/ month)
1
5
10
30
50
90
agile
plan-driven
3
10
30
100
300
Size
(number of personnel)
On
AB
Off
AB
On
AS
Off
AS
Off
Uninvolved
70
50
30
10
Culture
(% thriving on chaos vs. order)
Client Involvement

Figure 2.4: ??????????

3

Case Study

3.1 Problem

3.2 Company F

For the validation of the OPS Framework company F was willing to participate in order to measure how agile are its teams. 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 300% increase in the size of the development and QA departments resulting in the need for organizing better the development and release processes.

3.2.1 Methodology F

In general, company F does not follow a specific agile methodology, but rather a tailored mix of others which suit the needs of its team.

Methodology F, as we can name it, embraces the following practices from the various agile methodologies [10], some of the them in bigger and some of them in a smaller extent.

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

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

Table 3.1: Practices embraced in method F

3.2.2 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.

Team Size	7
Roles	Team Leader (1) Developers (4) Testers (3)
Development Process	Method A
Area	Mobile
Tools used	Perforce Titanium
Iteration length	2-3 weeks

Table 3.2: Team A - Profile

Team Size	8
Roles	Team Leader (1) Developers (5) Testers (2)
Development Process	Method B
Area	Java
Tools used	Perforce Eclipse IDE
Iteration length	2-3 weeks

Table 3.3: Team B - Profile

Team Size	4
Roles	Team Leader (1) Developers (1) Testers (2)
Development Process	Method C
Area	Java
Tools used	Perforce Eclipse IDE
Iteration length	2-3 weeks

Table 3.4: Team C - Profile

Team Size	17
Roles	Team Leader (1) Developers (9) Testers (7)
Development Process	Method D
Area	Java
Tools used	Perforce Eclipse IDE
Iteration length	2-4 weeks

Table 3.5: Team D - Profile

3.3 Method

3.3.1 Introduction

In order to measure the adequacy, the capability and the effectiveness of methodology F, the described method by Soundararajan [15] was followed.

3.3.2 Assessing the Adequacy

Based on the descriptions
Initially the objectives fulfilled by methodology F were identified. As it can be seen in Figure 3.1 all five objectives mandated by the OPS Framework are followed.

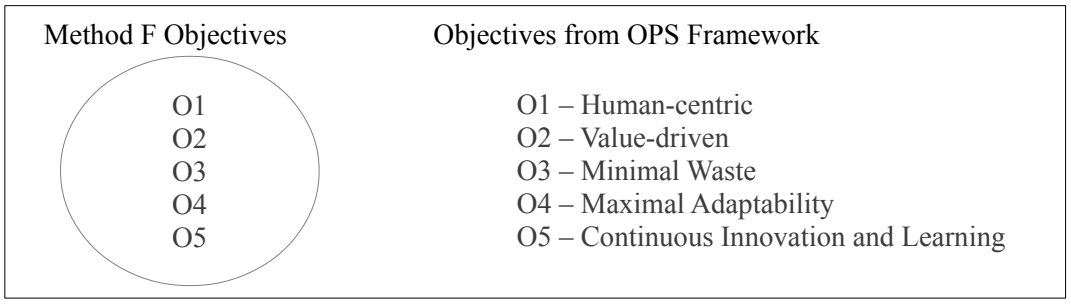

Figure 3.1: Objectives identified in methodology F

Based on the objectives and following the linkages from them the principles were identified. As one can see in Figure 3.1 methodology F does not follow the “Frequent Reflection

and Improvement" principle because the organization rarely does it re-examine the development process in order to improve it. It is worth mentioning that the "Empowering teams of Motivated Individuals" principles is not entirely followed, but it differs among the teams. Every team is built with motivated individuals, some to a more and some to a lesser extent.

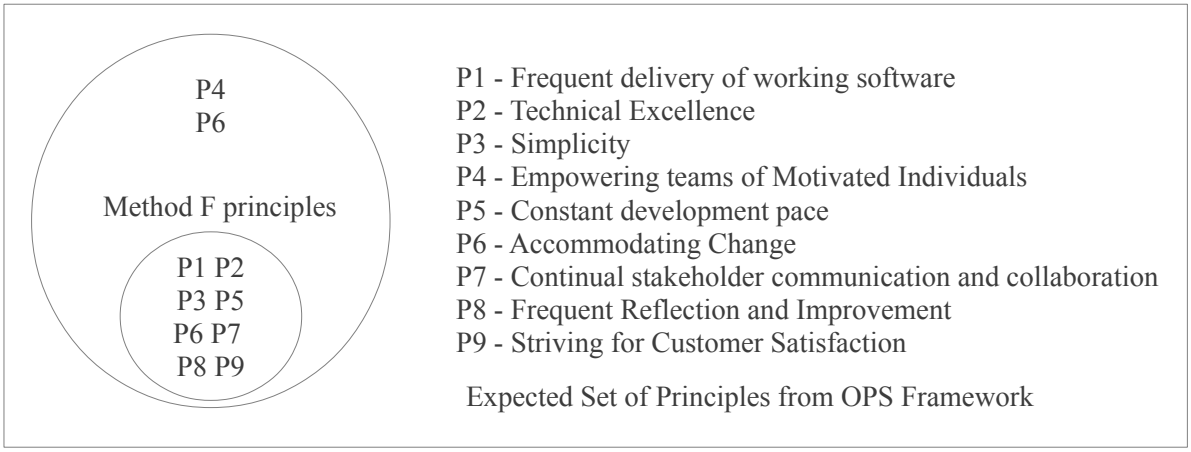

Figure 3.2: Principles identified in methodology F

Following the linkages from the principles the strategies for implementing them were identified. As it can be seen in Figure 3.3 methodology F does not support

- **Continuous feedback** - The organization does not have a defined process for getting a feedback from the customers of the company. From time to time the managers of the various departments of the company have personal conversations with the customers. If any issue arises, then they inform the development and QA departments in order to identify the problem and fix it.
- **Test-first development** - None of the team members writes tests before starting coding.
- **Constant velocity** - The organization does measure the velocity of the teams. In general the pace of development and integration and deployment is based on the needs of the customers and the capability of the developer. No one has to finish a specific amount of work in each iteration. What is only wanted, is the functionality to be ready when it is scheduled.
- **Retrospection** - Although there is a tendency to change this, the teams do not have a process for retrospection. The team members unconsciously consider that they are doing fine, unless a team leader or the manager of the department tells them the opposite.

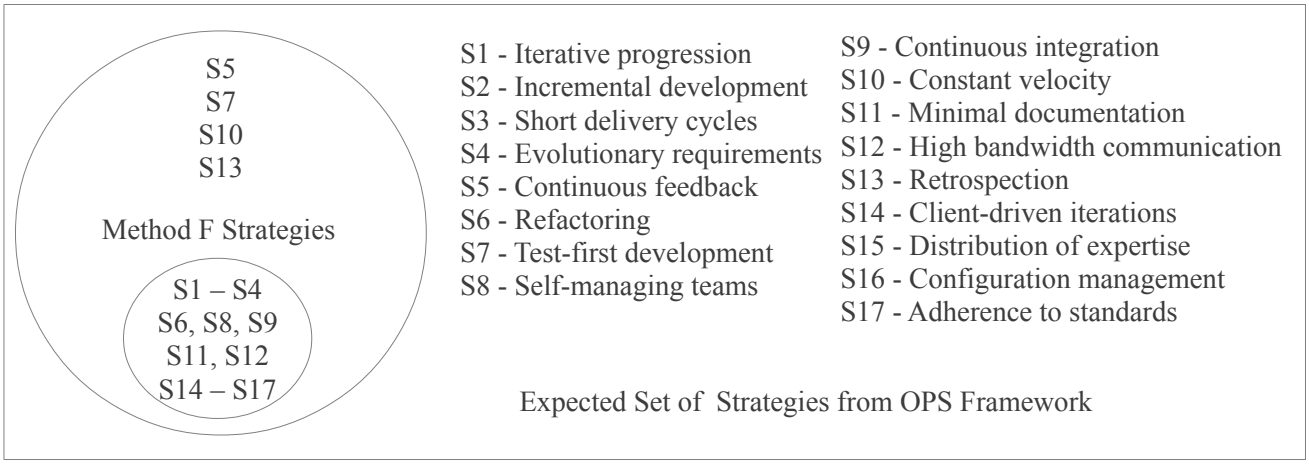

Figure 3.3: Strategies identified in methodology F

3.3.3 Assessing the Capability

3.3.4 Assessing the Effectiveness

3.4 Results

3.5 Discussion

Appendices

A

The Capability and Effectiveness Hierarchy for Company F

A.1 Capability Hierarchy

- 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?
    - 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?
- 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?
- 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?
  - Bug/Enhancement Completeness
    - To what extent has each bug/enhancement been coded?
    - To what extent has each bug/enhancement been unit tested?
    - To what extent has each bug/enhancement been refactored?
    - To what extent has each bug/enhancement been checked into the code base?
    - To what extent has each bug/enhancement been integrated with the existing code base?
    - To what extent has each bug/enhancement been reviewed?
    - To what extent has each bug/enhancement 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 bug/enhancement?
  - 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 list maintained?
    * To what extent are the bugs/enhancements fully estimated when added to the list?
    * To what extent are the bug/enhancement prioritized when added to the list?

- 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?

A.2 Effectiveness Hierarchy

B

Effectiveness Questions for Company F's Teams

To what rate are the following implemented?

1. Iterative Progression (Develop the product over several iterations/cycles in sequence)
2. Incremental Development (Create the product incrementally. Develop only a selected/prioritized set of bugs/enhancements during a release cycle)
3. Short Delivery Cycles (Deliver valuable software frequently)
4. Evolutionary Requirements (Allow the features/requirements to evolve over the development lifecycle)
5. Refactoring (Refine the architecture, design, code, and/or other process artifacts regularly to improve the quality of that artifact)
6. Self-Managing Teams (Allow the team members to determine, plan, and manage their day-to-day activities and duties under reduced or no supervision)
7. Continuous Integration (Team members integrate their work frequently; usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible)
8. Minimal Documentation (Maintain just-enough documentation to satisfy the needs of the development team and the customer)
9. High-bandwidth communication (Facilitate continuous communication among the (developers, testers, customers) (in-person, face-to-face interactions))

10. Client-driven iterations (The customers and users prioritize the bugs/enhancements. Build only what is of value to the customers and users)
11. Appropriate distribution of expertise (Select the right people to complete the tasks. Ensure that the team is composed of people with the appropriate skill sets to complete the assigned tasks)
12. Configuration Management (Manage the evolution of the product and other artifacts, both during the initial stages of development and during all stages of maintenance)
13. Adherence to Standards (Conform to a set of standards that the team or organization has agreed to comply with, e.g. Coding standards.)

Bibliography

[1] 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.

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

[3] 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/`

[4] 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.

[5] 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.

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

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

[8] 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.

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

[10] 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`

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

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

[13] Poppendieck, M., Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

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

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

[16] 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: Münch, J., Vierimaa, M. (Eds.), Product-Focused Software Process Improvement. Vol. 4034 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 290–304.
URL `http://dx.doi.org/10.1007/11767718_25`