Leiden University

ICT in Business

Measuring agile projects for portfolio management using project outcome

Name: James Lo
Student-no: 0756105

Date: 29/07/2013

1st supervisor: dr. M.R.V. Chaudron
2nd supervisor: drs. C.J. Stettina

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ABSTRACT

Background: The framework of McLeod and MacDonell (2011) describes the factors influencing the outcomes of a project. The model consists of four dimensions: People and Action, Development Processes, Project Content and Institutional Context. This thesis connects the factors described to be affecting the outcomes of a software system development project with factors of an agile project to come up with a measurement tool for program and/or portfolio management. **Method:** Questionnaires were constructed to measure the dimensions People and Action and Development Processes. Interviews were held with project managers to get insights into Project Content. Portfolio or program managers were interviewed to determine subjects for reporting and to discover challenges in this area. **Results:** Strong associations were found between People and Action (e.g. knowledge growth, coordination, open communication and motivation) and project outcome, but also between Development processes (e.g. Coding Standards, Embracing changing requirements, Customer visible/valued features and Planning poker) and project outcome. **Conclusion:** The People and Action dimension can be measured by the measurement tool used. Taking applications of agile practices as a measurement to determine the Development Processes requires further research. Project content could not be determined to affect project outcome. Portfolio management is mainly interested in factors from the Project Content dimension. However, all dimensions are relevant to measure for Portfolio management.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Project success within systems development is difficult to measure as it consist of multiple factors such as: technical, economic, behavioural, psychological and political dimensions (McLeod & MacDonell, 2011). A study of DeLone and McLean (2003) describes product success to be defined in system quality, information quality, services quality, use (or intention to use), user satisfaction and net benefit. McLeod and MacDonell (2011) show that project outcome, where project success is a derivative of, can be measured by the high quality of the product and development processes. Project success is found to be difficult to be labelled, because different studies present different factors that determine the outcome of the project (McLeod & MacDonell, 2011). On the other hand, organisational success is partially determined by the projects or moreover the portfolio that is held by the organisation. Handling the portfolio and programs on strategic and operational level will support the success of the organisation. It is therefore required to monitor projects in order to ensure project success (Krebs, 2008).

Different software development methods exist in order to effectively finish projects with a positive project outcome. Agile Software Development (ASD) is an incremental software development method, which nowadays is increasing in popularity throughout organizations (Dybå & Dingsøyr, 2008). The reason to migrate to this development method is that other development methods such as Waterfall and Rational Unified Process (RUP) contain a large part of specification and documentation (Ambler, 2005). Typical errors follow from misunderstandings in requirements, which often lead to extra development measures. Boehm (2002) describes the different home grounds of ASD and Plan driven approaches. ASD focuses more on the implementation and frequent communication with the customer. Agility is about trusting in one's ability to respond to unpredictable events more than trusting in one's ability to plan ahead for them (Fowler & Highsmith, 2001). Therefore also the people factor is as important in a project (Cockburn & Highsmith, 2001).

With the transition to ASD questions could arise involving the productivity of building the software, as ASD is also putting the focus on the people factor instead of processes. Metrics are measured to track the success of the project. Moreover, questions

surrounding the advantages spread around by ASD are discussion points. A more complete view of what factors will affect a project outcome will be further investigated in this study.

The current research question arose within a large organisation – which will not be mentioned by name due to confidentiality – that develops medical-related software. In this company a switch has been made from working with the Waterfall method to working with ASD, more specific Scrum. Scrum is an agile software development method focussing on project management that works as a wrapper around existing engineering practices to iteratively and incrementally develop software. As some projects do not perform better and/or software is not faster delivered with agile, the Waterfall method is still used (Williams, Brown, Meltzer, & Nagappan, n.d.). The focus for the current study will be on the factors in the development process of the projects that only use agile as development method. The literature study shows which metrics are compatible with agile projects without suffering its performance; however it will not go deeper into the subject.

McLeod and MacDonells model (2011) describes and identifies how the project outcomes are influenced by a number of influential factors within 4 domains: Instiutional Context, People & Action, Development Process and Project Content. These factors were found in multiple studies to affect project outcomes in systems development. Based on the dimensions a measurement system is attempted to be build.

1.1 RESEARCH QUESTION

The main question of this study is: How can the model of McLeod and MacDonell (2011) be operationalized in metrics for agile software development projects and programs/portfolios? Companies have introduced agile software development as a new method of building software. The metrics used for agile software development are similar to other software development methods to a certain extent. However, the focus of agile, which will be discussed in section 2, is different.

1.2 SIGNIFICANCE TO KNOWLEDGE

Studies have focussed on the metrics used by ASD for project management, but little for reporting to portfolio- or program management. These metrics measure specific attributes in the development process. This study contributes to finding a measurement tool to measure project outcome in terms of influential factors, such as the people factor. Next to this, the current study also contributes to the research field of agile software development and portfolio management, by looking how and if these factors associates.

1.3 OUTLINE OF THIS THESIS

The remainder of this thesis is given below.

Section 2	In section 2 a literature review is performed on the topic agile and software metrics in the field of program and portfolio management. It includes descriptions and research done on agile software development, scrum and portfolio management. The metrics that are found in the literature will also be explained.
Section 3	In this section the research strategy, methods and design are described. Furthermore, the approach of data gathering and analysis strategy will be covered.
Section 4	Section 4 will contain the results following from the activities (i.e. questionnaires and interviews) mentioned in section 3.
Section 5	The discussions of this study is found in section 5
Section 6	Recommendations regarding the use and follow up of the thesis will be given in section 6.
Section 7	Section 7 contains the conclusions of this thesis.

2 LITERATURE REVIEW

In this section the state-of-the-art coveres the areas that are related to the main question. The main subjects that are in the question are *project outcome* and *agile software development*. Agile Software Development (ASD) is described by finding research concerning the history of ASD, its definition, usage, benefits, concerns and metrics. As ASD is a software development method, also literature will be discussed that deals with alternative software development methods. Project outcome will cover literature other than that of McLeod and MacDonell (2011). As this study attempts to suggest for a measurement system, the metrics are discussed both on project level and program/portfolio management level.

For the literature study several libraries were used in order to find articles that cover software metrics and/or agile metrics. Furthermore, the state-of-the-art surrounding the topics of agile and scrum were searched using these libraries. The search terms used in these libraries were the terms “software”, “agile”, “agility”, “scrum”, “metrics”, “software development” and “measurement” combined. Next to the metrics, the state-of-the-art covering program- and portfolio management and project successes were gathered using the same libraries or collections. The sources that were used for this search were:

- Science Direct
- Wiley InterScience
- IEEE Digital Library Computer Science
- ACM Digital Library
- Springerlink
- Google Scholar

The articles that are used describe research that has been done in the field of agile software development or the agile method scrum. To link the topic metrics to the operation of the framework of McLeod and MacDonell (2011), the subject program and portfolio management are included. The metrics are a part of the framework to report to program or portfolio management.

2.1 SOFTWARE DEVELOPMENT METHODS

In the area of software engineering multiple software development models exist. These models have been made such that high quality software can be realised. A subset software models described in its general form are (Munassar & Govardhan, 2010):

- Waterfall model: Separate and distinct phases of specification and development
- Incremental model: pieces are added each time
- Iterative model: The distinct phases of the waterfall method are repeated

The iterative and incremental development models combined, form the basis of ASD (Larman & Basili, 2003). ASD will be more thoroughly described in the next section. Other software development models will not be further elaborated in this thesis except for the Waterfall method in the next paragraph. This is seen as the complement of ASD (Boehm, 2002).

A commonly used software development model is the Waterfall method, which is depicted in Figure 1. The Waterfall method is a method that can be categorised as what is called a plan-driven method. The plan-driven approaches of software development have been defined as document-driven, code-driven, and traditional process models (Boehm, 1988). This method of software development shows a linear flow of the processes, with an emphasis on: planning, time schedules, target dates, budgets and implementation of an entire system after complete specification.

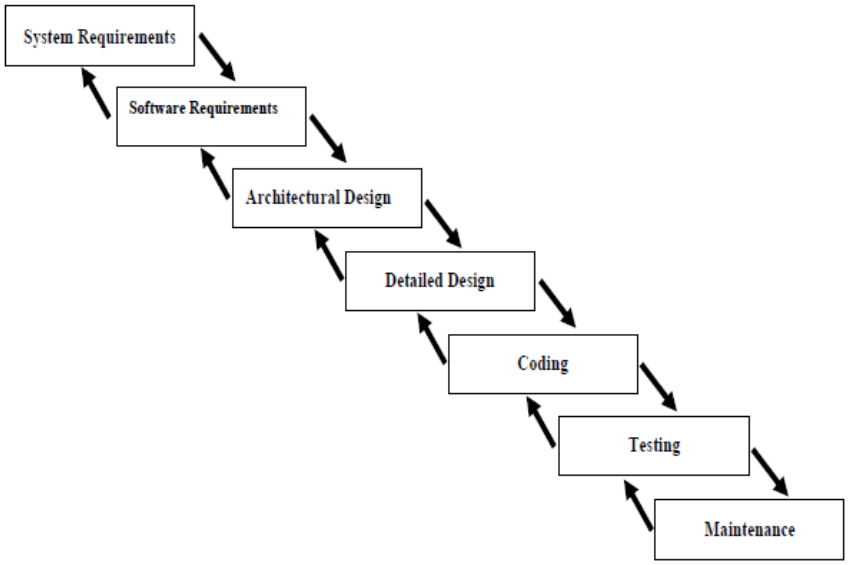


Figure 1. The waterfall method (National Instruments Coorporation, 2006)

In theory every phase needs to be completed before the next phase starts. This gives a good structure to the project, but can be costly when changes need to be made in the later part of a project (Ward-Dutton, 2011).

In the current environment changes come faster than in the past, in which the plan-driven approach has the downside of making concrete planning in heavy weight architecture and design. Therefore, this approach lack in coping with changing requirements (Boehm, 2002). Large projects are in that sense not different and also subjected to changing requirements. However, it can be argued whether an agile approach helps coping the large projects in general (Ambler, 2006; Boehm, 2002). The choice of software development method impacts the duration and cost of the project and in general the project outcome (McLeod & MacDonell, 2011). In order to get a better understanding of what agile is, it is described in this section.

2.1.1 Agile Software Development

The amount of different software development methodologies that were applied prior to ASD varied to a great extent. The development and promotion for ASD came from practitioners and consultants (Conboy, 2009). The most straight forward approach software development method, the Waterfall method, has been predominantly used throughout the decades, although incremental development already existed as far as

1930's. However, until the latter part of the 1990's the awareness of ASD only significantly accelerated (Larman & Basili, 2003). The history of ASD goes as far back as the mid-1950s (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003; Larman & Basili, 2003). In 2001 the Agile Manifesto (Fowler & Highsmith, 2001) was created by the Agile Alliance, which contained a set of purposes and principles when practicing Agile. The values stated in the Agile manifesto consist of:

> *Individuals and interactions over processes and tools*. This value insists that individuals and interactions should be valued more than processes and tools. If the individuals in a project are competent enough they will be able to use any process in order to obtain their goals. A lack of competency cannot per se be overcome by processes (Cockburn & Highsmith, 2001).
>
> *Working software over comprehensive documentation*. The focus is more on the quality of the software than the specification of the software. However, in situations where the future is predictable it will disadvantage the architecture that could have been set up prior (Boehm, 2002). By setting up the specifications costs, time and requirements will be different from the requirements at the start, focussing on working software will be more effective.
>
> *Customer collaboration over contract negotiation*. The manifesto also mentions customer collaboration to be of importance. An agile methodology that ignored customer collaboration and incremental development would almost certainly fail (Paulk, 2002). A method to enhance this purpose is to involve agile contracts (Sutherland, 2008).
>
> *Responding to change over following a plan*. In the time that plan-driven methodologies deliver output to the customer requirements could have changed. By having iterations these changes can be resolved earlier.

A study looking into the agile principles, found in Appendix B: 12 principles of the Agile Manifesto, and what applications of these are, are found to fit a more concrete and fundamental definition of the agile software development method in practice (Conboy, 2009). Some principles were found redundant or not concise. In Appendix B the refined agile principles can be found that are based on comments received from professionals on

the principles in the survey (Williams, 2012). Also the definition of agile is used differently throughout the literature. A redefined definition of agile based on theory is (Conboy & Fitzgerald, 2004):

> *"The continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through its collective components or its relationships with its environment"*

The definition of Conboy & Fitzgerald (2004) has been altered in a later publication to add the learning aspect when changes occur. They modified the definition of agility in *Information System Development* (ISD) to:

> *"The continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change, through its collective components and relationships with its environment."*

Agile in large organizations is not commonly adopted. In a survey by VersionOne, 13% of the respondents working at larger organizations (more than 500 employees) said that nearly all their projects used agile (VersionOne, 2011). Agile is a reaction against traditional methodologies, also known as rigorous or plan-driven methodologies (Boehm, 2002). This can be troublesome in large organization, where it is difficult to assign customers or a proxy to self-organising teams (Kettunen, 2007). Some large organisations describe how they have implemented ASD in their organisation, such as Yahoo!, OLCL and Microsoft. However, the paths that the organisations followed were not the same, where a reason can be the different organisational constructs or culture (Benefield, 2008; Miller & Carter, 2007; Tudor & Walter, 2006). According to a study of Forrester (2011) organisations fall back or into Water-Scrum-Fall where processes of the Waterfall method and Scrum are mixed. This should not cause problems in all cases, but negative effects can occur, e.g. when too much time is spent on writing many early requirements resulting in too many wrong requirements (West, Gilpin, Grant, & Anderson, 2011).

It is different to implement agile in large organisations compared to implementing it in small to medium sized organisations. The article of West et al. (2011) also refers to an article mentioning that the applicability of the agile software development method is often found challenging in large organisations. To overcome the challenge of applying agile, a

mix of agile and traditional elements of software development must be set up. Mainly large organisations require a plan-driven approach in order to plan and document for reporting the process. There has to be decided where to place the balance between documentation and planning while achieving the flexibility and benefits (Paulk, 2002). When combined, it might create a better method fitting certain organisations. A mix of plan-driven and agile methods can provide a combination of properties that include the best of both worlds (Boehm & Turner, 2005). The two development methods can be sustained next to each other, with obstacles, thereby creating an ambidextrous organisation (Vinekar, Slinkman, & Nerur, 2006).

Agile methods work best when customers operate close in collaboration with the development team and when their tacit knowledge, knowledge that is difficult to transfer, is sufficient for the full span of the application (Boehm, 2002). Plan-driven methods is said to reduce this risk by the use of documentation and architecture review boards. However, the linear nature of the Waterfall method is its largest problem. The process does not define how to respond to unexpected changes from any intermediate process (Schwaber, 1994). Boehm (2002) also mentions that plan-driven methods work best when developers can determine the requirements in advance and when the requirements remain relatively stable with change rates on the order of one per cent per month. It describes the home-grounds of both methods shown in Table 1. Both of the software development methods have their own strengths and weaknesses, therefore varying in success in different situations.

Differences Agile methods and plan-driven methods		
Home-ground area	**Agile methods**	**Plan-driven methods**
Developers	Agile, knowledgeable, collocated, and collaborative	Plan-oriented; adequate skills; access to external knowledge
Customers	Dedicated, knowledgeable, collocated, collaborative, representative, and empowered	Access to knowledgeable, collaborative, representative, empowered customers

Requirements	Largely emergent, rapid change	Knowable early; largely stable
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams and products	Larger teams and products
Primary objective	Rapid value	High assurance

Table 1. Differences between agile methods and plan-driven methods (Boehm, 2002)

There are several agile methods: i.e. eXtreme Programming (Beck, 1999), Scrum (Schwaber & Beedle, 2002), Feature-driven Development (Palmer & Felsing, 2001), Test-driven Development (Beck, 2002) and Leagile (Wang, Conboy, & Cawley, 2012). A systematic review shows the most relevant studies in the area of Agile, where it found that agile in general and eXtreme Programming (XP) were mostly studied in the literature and that the Scrum method is still requiring attention for research (Dybå & Dingsøyr, 2008). A follow up systematic review calls for the need of researches with theoretical roots (Dingsøyr, Nerur, Balijepally, & Moe, 2012). It also acknowledges the need for research of agile methods and practices in different contexts. The survey of VersionOne that was mentioned earlier shows that 52% of the participants uses scrum as an agile method (VersionOne, 2011). This method is mostly seen as the agile method to introduce in larger organisations.

2.1.2 Scrum – An agile method

Scrum is a framework based on the principles of ASD originating from practice. It is a development management, enhancement and maintenance method for an existing system (Schwaber, 1994). Of the agile methods, Scrum is the most used agile method used by organisations (VersionOne, 2011). In the figure below the Scrum process is given, consisting of the *pregame*, *development* and *postgame* phase (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). The key roles, events and artefacts that are used within Scrum are further elaborated in this section.

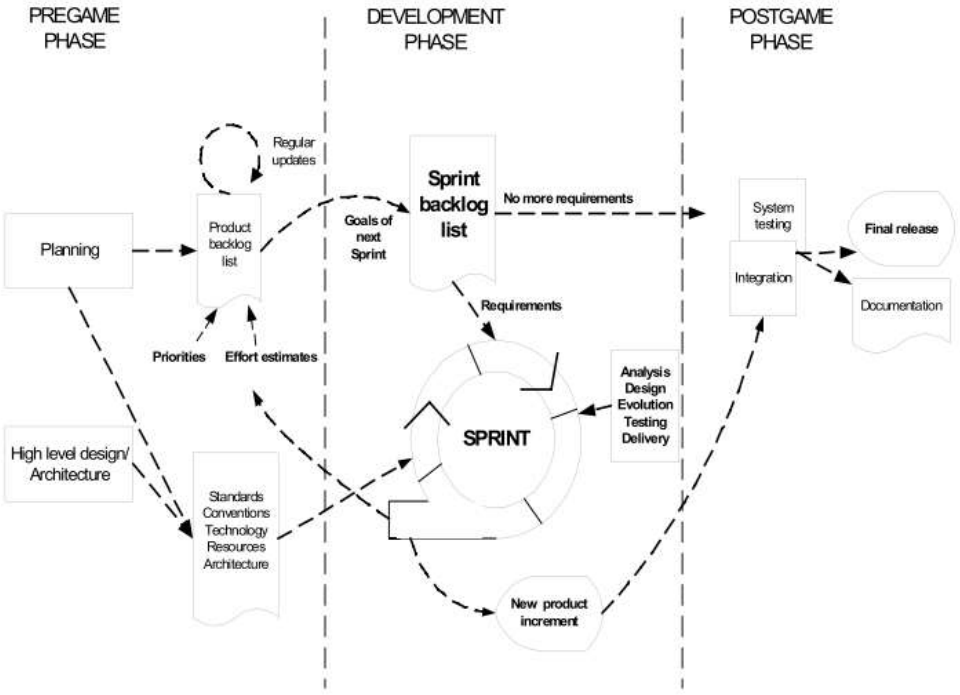


Figure 2. Scrum process (Abrahamsson et al., 2002)

The key roles in the scrum team are found below (Sutherland & Schwaber, 2011):

Product owner (PO) – ensures that the end product conforms to the requirements and has maximal value. Furthermore, the PO is responsible for the development team and moreover the product. In order to optimise the Product Management process, where a PO for example is taking care of the Product Backlog, Scrum principles can also be applied (Vlaanderen, Jansen, Brinkkemper, & Jaspers, 2011).

Scrum master – ensures that the Scrum team, consisting of product owner, development team and scrum master, follows the Scrum theory, practices and rules. Also the scrum master finds ways to improve the scrum process. In large organisations multiple teams could arise that work on the same project. With each team having one or a shared scrum master it is necessary to align the teams. This can be done by *Scrum of scrums*, where Scrum masters meet up to discuss day-to-day developments (Sutherland, Viktorov, Blount, & Puntikov, 2007).

Development team – creates the software according to the specifications set by the PO. The team will make estimations (e.g. story points on user stories) and deliver a user story at each release if it is "done". What done means, will be explained in the description of the *Sprint backlog*.

Key events in Scrum are:

Sprint Planning – is an event where the scrum team collaborates to determine the how and what will be delivered in the upcoming sprint.

Sprint – a cycle where the development team builds the functionality selected from the Product Backlog placed onto the Sprint Backlog.

Daily Scrum Meeting / Stand-ups – are meetings where each individual of the development team informs the team of his/her activities for the coming day. Usually the stand-ups are daily to update the team of current progression. There is more to standing-up (Yip, 2011):

- To help start the day well
- To support improvement
- To reinforce focus on the right things
- To reinforce the sense of team
- To communicate what is going on

Sprint Review – occurs at the end of a sprint where the scrum team and stakeholders discusses (and demonstrate) what have and/or have not been achieved in the sprint.

Sprint Retrospective – takes place after each sprint, where in general the lessons learned are discussed (Kerth, 2000).

Key artefacts in Scrum:

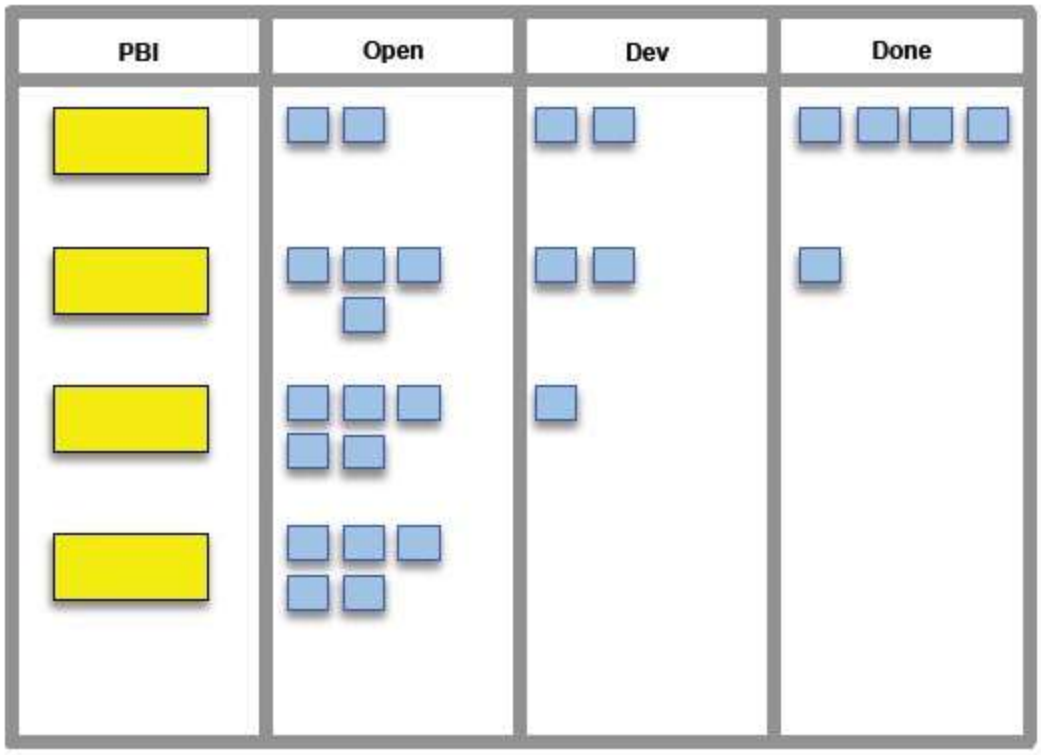
Product backlog – is a prioritized list of functionalities or non functionals of a product that needs to be implemented by the development team. Prioritizing the backlog is suggested to be based on business value for the organisation (Hundermark, 2009). Based on the priority and estimations, usually the velocity of

a team, the functionality where the development team will be working on in the upcoming sprint will be determined by the PO in collaboration with the development team.

Sprint Backlog – After choosing the functionalities of the product backlog it will be placed on the Sprint backlog. On this board, the development team will find the tasks they will be working on. In general there are three categories: open, in progress and done. The *definition of done* is a term used in Scrum to indicate when a task is done and is defined by the team. A more mature way of defining the *definition of done* is to use Quality Assurance Schedule (Jakobsen & Johnson, 2008). An example of what a QAS document can contain is:

- What stories are subject to inspection
- What code is subject to review
- What documents are subject to what types of review
- What unit test and automatic test is produced
- What is included in the acceptance test

Tasks found in *open* are not currently worked on by the team. The tasks in *dev* (see below) are currently being worked on.



(Source: http://www.methodsandtools.com/archive/atddreadysprintbacklog.php)

Figure 3. Example of a sprint backlog

A non-Scrum event *Story Point Poker (Planning poker)* is a method to estimate the story points of a user story and to reduce the chance that a user story will be under or over evaluated (Cohn, 2005; Grenning, 2002). A study conducted by (Mahnič & Hovelja, 2012) found that the group discussions helped improving the accuracy of estimating story points.

Retrospective initiates the establishment of a learning culture by identifying issues and seeing the importance of improving these issues (Kerth, 2000). Project success and failure can be qualitatively analysed by use of a Retrospective (Nelson, 2005). It gives information that is useful from several perspectives and benefits by having: organisational learning, continuous improvement, better estimating and scheduling, team building and improved recognition and reflection.

2.1.3 Value individuals over processes

The agile manifesto values the focus to be shifted more to the people than on the process. This however, does not mean to neglect the process. In a self-organising team the individual is important for its team performance (Cockburn & Highsmith, 2001).

"*To have a team with the right expertise – good domain experts, good developers, good chief programmers – will make no process make up for a lack of talent and skill.*" (Cockburn et al., 2002, p 6)

A study confirming the Critical Success Factors (CSF) of agile projects also found one of the CSFs to be the capability of the team (Chow & Cao, 2008). Other CSFs found in the same study are factors that rely on processes. Studies that attempt to assess the factors in the area of the *people factor* focus on several areas. Psychometrical measures in the area of agile is gaining interest (So & Scholl, 2009). A measure / tool for agile teams to self-reflect is one of these studies (Stettina & Heijstek, 2011). Here five agile factors are being assessed, namely: leadership, learning, redundancy, autonomy and team orientation. Also Krebs (2008) captures *team morale* in his factors to be measured for agile projects. This will be discussed further in section 2.3.1. A lack of face-to-face meeting, for example, is found to be a cause for the lack of sense of team cohesion, which creates a barrier to build trust within the team (Iqbal & Abbas, 2011). A lack of trust between Scrum master and team members was found to be an important reason for a team member not to report problems and to give instructions to a team member what to do (Moe, Dingsøyr, & Dybå, 2010). Distributed software development teams can cope with not having a face-to-face meeting in their projects. A solution for connecting isolated Scrum teams is to let Scrum masters meet; a scrum-of-scrums (Sutherland et al., 2007). A face-to-face meeting is one of the practices that is mentioned in the Scrum method, being an effective method to transfer information (Fowler & Highsmith, 2001; Sutherland & Schwaber, 2011).

2.2 WHAT WE MEASURE TO IMPROVE PROJECT OUTCOMES

Organisations are always finding methods to have a better competitive performance / competitive advantage, e.g. by improving the ability to keep innovating (Michael E Porter,

2008). Switching from software development methods can be argued by criteria, such as reducing time-to-market comparing the waterfall method with an agile method (Huo, Verner, Zhu, & Babar, 2004). In order to not have setbacks in the process optimization, guides and assessments are made available. Metrics to measure certain activities within the project can give information that for example give the status/progress of a project. Immature organisations usually have processes made up by practitioners and managers, causing project managers to focus on fire fighting, because there is no objective basis for judging a product or solving conflicts (Amaratunga, Sarshar, & Baldry, 2002). Software Process Improvement (SPI) can help improving production and quality within time and budget (Arent & Nørbjerg, 2000). The most well-known model to indicate a mature company is the Capability Maturity Model Index (CMMI) (CMMI Product Team, 2006). This model can assess an organization and let it show where to be focusing on in order to reach a next level. CMMI describes 5 levels of maturity. Being more mature means processes are being more effective and efficient (Goldenson & Emam, 1995). A similar method focusing more on the processes occurring in a software development process is the Software Process Improvement and Capability dEtermination (SPICE, ISO/IEC 15004) (Paulk, Curtis, Weber, & Chrissis, 1993). By determining the performed level of a certain process the maturity of the process can then be assessed. Next to maturity, metrics can have various purposes.

2.2.1 Software metrics

The definition of a metric (Oxford Dictionaries, 2012):

> *“Metrics (in business) a set of figures or statistics that measure results.”*

Similar to engineering, software can be evaluated based on characteristics such as quality and cost etc. The definition of software metrics can be defined as (Goodman, 2004):

> *"The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products."*

The definition mentions information gathered in order to improve process and product. This information can be retrieved from several metrics and even more measurements. The

main categories of software metrics are relevant to software engineering are (Agarwa & Tayal, 2009):

1. *Product Metrics*. Product metrics describe the characteristics of the product such as:
    - Size
    - Complexity
    - Performance
    - Reliability
    - Portability
2. *Process Metrics*. Process Metrics describe the effectiveness and quality of the processes that produce the software product. For example:
    - Effort required in the process
    - Time to produce the product
    - Effectiveness of defect removal during development
    - Number of defects found during testing
    - Maturity of the process
3. *Project Metrics.* Project metrics describe the project characteristics and execution. For example:
    - Number of software developers
    - Staffing pattern over the life cycle of the software
    - Cost and schedule
    - Productivity

Within product metrics there are again two kinds of metrics (Scotto, Sillitti, Succi, & Vernazza, 2006):

- *Dynamic metrics* are collected by measurements during program execution. It can be useful for assessing the efficiency and the reliability of a program.
- *Static metrics* are collected by measurements based on the system representations such as design diagrams, source code, or documentation. It is useful to be able to understand the complexity, understandability, and maintainability of a software system.

A few commonly measured software measurements are used to calculate: schedule, size, cost and quality. Here schedule and costs are project metrics and size and quality are software metrics. The latter two can be calculated using several methods found in Table 2, which is an unexhausted list.

Size/Complexity	Quality
Function point analysis	Bugs per line of code
Cyclomatic complexity	Code coverage
Number of lines of code	Cohesion
Number of classes	Program execution time
	Program load time

Table 2. Examples of size and quality metrics

Function Point Analysis measures the complexity of the software. By counting the following criteria an objective result is retrieved: External Inputs (EIs), External Outputs (EOs), External Inquiries (EQs), Internal Logical Files (ILFs) and External Interface Files (EIFs). The first three are treated as Transactional Function Types and last two are called Data Function Types (Gollapudi, 2004).

Cyclomatic complexity measures the complexity of the software. This is done by measuring the independent paths in the code. A practical measurement that can be performed is to compare the number of tested paths against the cyclomatic number. This will indicate whether: more test should be run, actual paths can be reduced or program size can be reduced (McCabe, 1976).

Lines of code and *Number of classes* are both used to measure the size of the software. It is used to estimate the amount of effort that is required to develop a program or to estimate the productivity of a finished program (Gollapudi, 2004).

2.2.2 Agile metrics

In the literature many of the agile metrics contains measures that are used for measuring the progress of the project. Here the use of metrics should reflect on the economic and financial aspects (Anderson & Schragenheim, 2003). Also, some measurement methods do not match with the principles of agile. A few project and product metrics and measurements used within agile projects are described here.

Velocity is the most commonly used metric in an agile project. It is determined by the team by estimating how much it thinks it can produce in a iteration (Cohn, 2004). When the

velocity is known, the Product Owner plans the user stories basing it on the velocity of the team. Care should be given when estimating based on the velocity as it can be affected by multiple factors. A few examples are: changing team members, obstacles, toolsets, difficulty of feature or amount of learning required, etc. When there are no obstacles for the team get stable and in general have an increase in velocity (Hartmann & Dymond, 2006).

Based on the principle of the velocity, *Agile Earned Value Management (AgileEVM)* is a metric translated from Earned Value Management that is to relate to the Scrum methodology (Sulaiman, Barton, & Blackburn, 2006). It is measured by calculating the Earned Values against the Planned Value of an entire release and therefore shows an integration of scope, schedule, resources. Methods are shown how to put Earned Value Management in practice (see also Cabri & Griffiths, 2006). To report progress to executive management measures graphical representations will quickly indicate, for example, a status of a project (Barton et al., 2005). Next to measuring AgileEVM for progress it can also be measured for project performance.

A report based on a number of podcast episodes of IBM summarizes six measurements, shortly described below, to focus on to keep a sustained improvement in software productivity. It places the focus on progress metrics and quality metrics. These metrics are not specified for agile methods, but can be applied to it. Quality metrics, mentioned here, overlaps with product metrics that is described before. The three progress metrics are: *Planning progress*, *Technical progress* and *Economic progress*. Planning progress in an agile project refers to the planning of user stories. The technical progress can be summarized as learning from the mistakes/difficulties. The economic progress measures the estimation of costs until completion. The quality metrics are: *defects and changes due to defects*, *the costs of change* and *scrap and rework*. The first two quality metrics refer to the maintainability and adaptability of the software, which can affect the costs of the software in a later time period. The latter metric is to see what redundancies are found in the development process. This should be acknowledged and removed (Ward-Dutton, 2011).

A way to measure progress is to use the velocities of the teams to indicate it. Another method to measure progress is by using *System Instability metric* (SDI) which is

compatible with ASD (Alshayeb & Li, 2004). This metric measures the complexity of the software. The SDI metric measures the system-level design changes. The changes in class names and inheritance hierarchy, addition of new classes, and deletion of existing classes all indicate the stability/instability of a design and the abstraction of an application domain and basing on the changes indicating the progress of the project.

Code coverage is a measure used in testing to assure the quality of the software. It measures the percentage of code that will be covered during the tests. This is a common measure not specific for agile, however compatible.

As agile include people to be a measure of success a method, *team morale* should be measured (Krebs, 2008). This will be further described in Portfolio Management. A similar measure is the happiness metrics, but morale is a more general measure also including the happiness of an individual.

Scrum teams have a different focus than development teams working with a software development method different from agile. In order to measure the performance of scrum teams therefore need other measurements to get results for certain metrics. Downey and Sutherland (2012) describe nine metrics to develop and sustain *Hyperproductive Teams*. A Hyperproductive team is a team that performs at least 400% better than an average team developing based on the waterfall method. The nine metrics are needed, because without the metrics the performance of the team might get unstable and loses control which will result in a lower velocity (Downey & Sutherland, 2012).

2.2.3 Metrics discussion

With the focus of agile and scrum being people and delivering working software, often the metrics measuring the performance of an agile or scrum project will be different to measuring certain metrics in development methods, such as a plan-driven method. An example is: measuring the productivity of an individual. As a scrum team is working as a team measuring individuals is not as essential as measuring the performance of a team in general. Of course there are arguments when this could be necessary. Even here measuring the velocity of a team, as a variable for measuring productivity, can be problematic (Vishwanath, 2012). Another method to objectively measure productivity is based on Function Point Analysis set out against the man hours or days used to produce

the code. Reasons not to use this method is because the functional size is changing each sprint instead of once in a one-time delivery method making it less interesting to measure (Krebs, 2008).

Measuring too many metrics also does not contribute to project success (Krebs, Kroll, & Richard, 2008). To have the right amount of metrics therefore is also a challenge. Therefore the metrics should give useful information by not only minimal effort, but also correct information that helps the development team to progress in their learning and reaching the objectives (Glazer, Anderson, Anderson, Konrad, & Shrum, 2008; Krebs et al., 2008). Krebs et al. (2008) argue ways to not fall for the 5 common pitfalls in metrics collection: *bloated metrics*, *the evil scorecard*, *lessons forgotten*, *forcing process* and *inconsistent sharing*.

An interrelation is found between agile metrics and traditional metrics to be the broad classification of a metric, but differances in the approaches to measure (S. Misra & Omorodion, 2011). Some agile metrics uses subjective measurements for measuring metrics such as progress (Cohn, 2004). Basing user stories on story points that are estimated by a team it cannot be compared between different teams. An objective measure can be introduced to connect these, such as hours or Function Points. Misra & Omorodion (2011) plead for standardisation of agile metrics. Another problem however is that metrics such as project metrics exist to inform stakeholders about the status of a project (Kerzner, 2011). As described earlier, agile has the focus on people and should therefore include this factor in future measurements.

2.3 AGILITY IN PROJECT ORGANISATIONS

The focus of program management goals are on improving the efficiency and effectiveness by better prioritization, planning and coordination in managing projects (Blomquist & Müller, 2006). In program management multiple projects related to each other are congregated to one program. A programs life cycle are based on (Blomquist & Müller, 2006):

- Formulation (e.g. evaluation of options)
- Organisation (e.g. strategy planning, selection of actions)

- Deployment (e.g. execution of actions in projects and support)
- Appraisal (e.g. assessment of the benefits, reviewing purpose and capability)
- Dissolution (e.g. reallocation of people and funds)

Program management connects projects with the similar topic, mostly products. Projects are therefore part of the larger program. Multiple programs and projects are then again part of a portfolio.

The determination of metrics that needed to be measured in an organisation can be done top-down. An example of a method is the Goal Question Metric Approach (GQM). It is based upon the assumption that for an organization to measure in a purposeful way it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals (Basili, Caldiera, & Rombach, n.d.). Another approach, which can be more high level, can be done by a SPICE or CMM assessment. The assessments respectively show the capability and the maturity of an organisation. Here CMM assesses the organisation as a whole and SPICE also focuses more on the implementation and institutionalization of specific processes; a process measure (Paulk, Konrad, & Garcia, 1995).

Prior projects did not use agile as software development method. However, in some cases it could be desirable to see how current projects perform in comparison to historical projects. In order to link these two, some common metrics needs to be used. Here productivity can be used as a metrics as it measures the efficiency of production (Dybå & Dingsøyr, 2008). Function points should not be the key metrics preferred to be used in agile, although in an application portfolio, function points could be used to derive maintenance costs and the cost of common enhancement projects for an existing system (Jones, 2010; Krebs, 2008). The project metrics described above are part of program management. However, portfolio management, next to project portfolio, also requires the knowledge of how the projects fit/follow the strategy.

When talking about new product development (NPD) within the field of software the agility of projects affect the way of working of in rest of the organisation. An example is the frequency of delivery of new parts or products that needs to be sold by the sales

department. In a certain way the organisation needs to turn more agile as well (Kettunen, 2007). Three project agility dimensions can be identified, which are business uncertainty, project constraints and technical uncertainty (see also Figure 4). The project agility dimenstions are advocated to be combined with the NPD to create an organisation and balance the agility with the projects held in the portfolio. This model can be found in Figure 5. NPD enterprise agility.

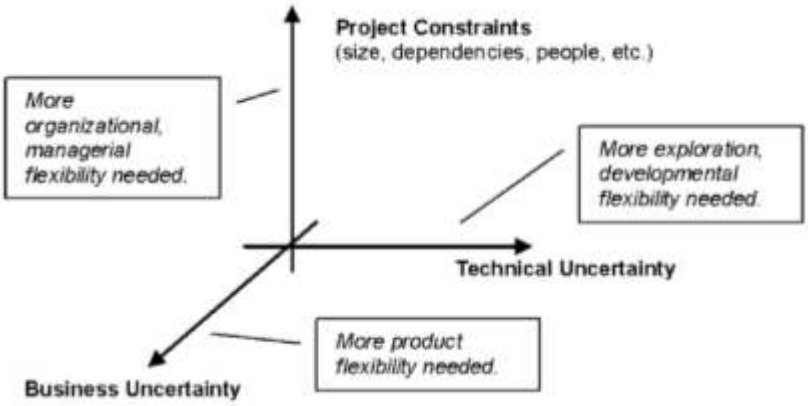


Figure 4. Project agility dimensions (Kettunen, 2007)

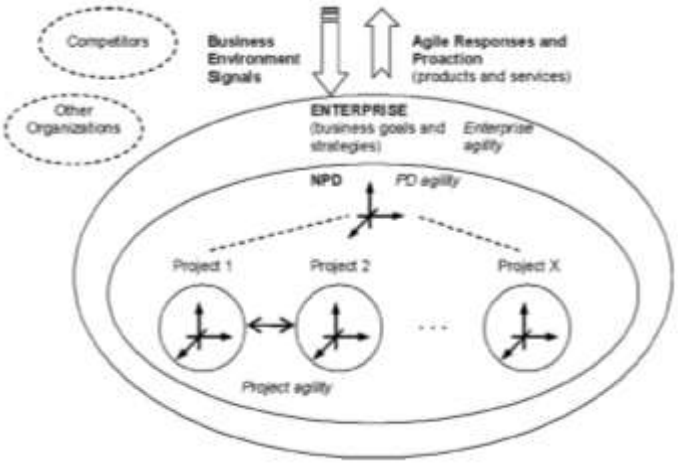


Figure 5. NPD enterprise agility (Kettunen, 2007)

The same article shows the most popular methods used – by organisations participating in their survey – to the projects or programs in order to analyse and evaluate the projects or

programs. The methods used are: financial methods, business strategy, bubble diagrams, scoring models and checklists. These multiple techniques can be used in a combination with each other for the assessment. Cooper, Edgett, & Kleinschmidt (1999) even suggest that using a combination of techniques that provide a general answer is better as none of the techniques.

2.3.1 Portfolio Management

Portfolio management consists of determining the strategy of the company by selecting which markets, products and technology a business needs to invest in (Cooper et al., 1999). The new product and technology choices that management makes will determine what the business will look like in a number of years. All projects or programs executed by a business fall under portfolio management and should be in line with the strategy of the organisation. Also, portfolio management is about resource allocation— the allocation of scarce and vital R&D, engineering, marketing, and operations resources at a time when these resources are more stretched than ever (Cooper et al., 1999). Doing too many projects with limited resources will results in longer cycle times, poor quality of execution, and underperforming new products. Portfolio management can therefore be categorized in asset-, project- and resource portfolio management in order to have the focus on personnel, project and existing systems (Krebs, 2008).

Several metrics to measure the performance of a portfolio are (Cooper et al., 1999, 343):

- “having the right number of projects in the portfolio for the resources available,
- avoiding pipeline gridlock in the portfolio—undertaking projects on time and in a time-efficient manner,
- having a portfolio of high-value projects (or maximizing the value of the portfolio)—profitable, high return projects with solid commercial prospects,
- having a balanced portfolio—long term versus short term, high risk versus low risk, and across markets and technologies,
- having a portfolio of projects that are aligned with the business’s strategy, and
- having a portfolio whose spending breakdown mirrors the business’s strategy and strategic priorities.”

Within portfolio management managing the projects relate to balancing the portfolio. Gerald Weinberg uses software engineering tasks as a basis for determining the productivity penalty and believes the number is much higher. Switching from one system to another system also negatively affects the productivity, as a software developer must memorize large pieces of the systems to understand dependencies and relationships. A switch between projects will therefore be a large shift. Weinberg predicts 20 per cent waste when switching between simultaneous projects (Krebs, 2008). It is therefore suggested that developers should not distributed over multiple projects at the same time.

2.3.2 Portfolio metrics

Showing the status of projects will inform stakeholders how the project and also resources and assets are performing (Kerzner, 2011). Krebs (2008) suggests three parameters for reporting metrics important of an agile project, namely: progress, quality and team morale. These metrics, which will be described more thoroughly below, can be measured with several measurement tools.

Measuring *progress* will show how the planned value compares to the actual value of a project. The result indicates whether if a project is running on schedule or not. Next to this, the results can be used as an estimation of the planned value of the next sprint. The commonly used method within Scrum is using story points. By comparing time to the number of story points “burned”, the state of the sprint can be expressed. The estimation of a next sprint can be based on the *velocity*, which can be calculated by using the amount of story points burned and man hours burned. Knowing the man hours or person-days available for the next sprint gives the number of story points that can be spend when choosing the user stories. Krebs (2008) mentions several other measurement methods, i.e. use-case points for organisations using use cases, COCOMO (Boehm, Clark, Horowitz, & Westland, 1995), Function Points (NESMA, 2011).

Quality is measured by looking at the number of unsolved defects of a project. Measurements to measure quality mentioned by Krebs (2008) are:

- Total number of defects,
- Ratio of total number of test cases to open defects
- Unit-test code coverage

- Total number of unit tests

Here Krebs advises to use the ratio of test cases to open defects as the measurement for quality, because it provides more information about the quality of the system.

Team morale measures the morale the team has at a certain point of time. Overtime has a negative impact on the productivity of a team. This follows due to the rise of the stress level. Other factors, however, can also influence the morale of the team. In longer projects a high turnover rate early in a project can show a negative impact in the later part of the project. Not many tools are available for measuring team morale within agile, the most well-known would be the happiness metric. This however only shows one variable, in a rating, asking an individual to what extent he/she was satisfied with the sprint. A more complete tool by means of a survey also can be used for the measurement of self-reflection (So, 2010).

It is not determined that one dimension of Development Process, People & Action or Project Content is determining the final outcome of a project. Therefore, reporting only the status of the project should not only contain information about the hard factors. In an agile project more emphasis should be put on the people factor (Cockburn & Highsmith, 2001).

2.3.3 Project success

Measuring the performance of aspects within the project has the common goal of reaching the objective of the projects and making it a successful project. In order to get the aspects to optimally execute a project it is also important to know what these aspects exactly are and why these are important. Process Improvement helps in minimizing overhead in and reducing the risk of project failure. Therefore maturing via CMMI or SPICE or any other maturing methodology will let the organisation thrive for project success. Metrics are to show how certain aspects in the organisation are performing (Kerzner, 2011).

A report of the Standish Group International (2009) shows how projects resulted in success or failure. Furthermore, they give a list of 10 success factors in order of importance:

1. User Involvement
2. Executive Support

3. Clear Business Objectives
4. Emotional Maturity
5. Optimization
6. Agile Process
7. Project Management Expertise
8. Skilled Resources
9. Execution
10. Tools and infrastructure

Looking further than the iron triangle (cost, time and quality) in project management, the success of a project consists of the factors mentioned in the *square route* found in Table 3 (Atkinson, 1999).

Iron Triangle	The information system	Benefits (organisation)	Benefits (stakeholder community)
Cost Quality Time	Maintainability Reliability Validity Information quality use	Improved efficiency Improved effectiveness Increased profits Strategic goals Organisational-learning Reduce waste	Satisfied users Social and environmental impact Personal development Professional learning Contractors profits Capital suppiers Content project team Economic impact to surrounding community

(Source: Atkinson, 1999)

Table 3. The square route to understanding success criteria

In the study of (Chow & Cao, 2008) 39 factors of affecting project success and 19 factors for project failure were found in literature. The project success were derived from case studies, meta-data, compilations and observations (Koch, 2005; Highsmith, 2002; Schatz & Abdelschafi, 2005; Karlstorm & Runeson, 2005; Augustine, Payne, Sencindiver, & Woodcock, 2005; Ambler, 2006; Reifer, Maurer, & Erdogmus, 2003; Lindvall, et al., 2004;

Boehm & Turner, 2005). Aspects of project failure are found in a number of studies (Cohn & Ford, 2003; Larman & Basili, 2003; Nerur, Mahapatra, & Mangalaraj, 2005; Reel, 1999; Boehm & Turner, 2005). The study itself found six Critical Success Factors (CSF) by means of a survey asking for the perception of success against the level of perception of the participant. The six CSFs, in order of importance, that were found were: *delivery strategy, agile software engineering techniques, team capabilities, project management process, team environment* and *customer involvement*.

2.4 FACTORS AFFECTING PROJECT OUTCOMES

The framework of McLeod and MacDonell (2011) describes 4 factors that influence the 5th factor, project outcome. Software development projects are usually judged as a successful project or a failed project. "*Success is a high-quality development process outcome and/or a high-quality product outcome*" However, these are difficult to quantify when trying to find the factors of these causes. The framework therefore describes that project outcomes are influenced by four dimensions. These dimensions are: People & Action, Development Processes, Project Content and Institutional Context. The same factors are found in the Figure 6, and also found in Appendix A: Framework of . Each factor consists of more specific properties, which also can be found in Appendix A: Framework of . Each property in every factor can contain the learnings for improvements in future projects. The domains and important factors will be further described in this part.

The qualitative model of McLeod and MacDonell (2011) describes multiple factors that influence the outcomes of system development projects. The dimensions and factors of the framework will be summarised here.

2.4.1 People and action

The dimension People and Action describes the important factors that are influenced by the human elements. The stakeholders within this domain again influence the project in its own way or function. Influencing the project can be conscious or subconsciously and affect the project with a certain impact. This impact however is not described. Project outcomes can be affected on several angles in the dimension of People and Action according to the model. As mentioned earlier, the most important roles in a scrum project

are: the scrum master, product owner and of course the developers. As the developers are creating the end-product, in collaboration with other roles, they are the main determinants of the success of a project. Here not only individual skills are an influential factor, but also the social interaction within the team.

Other stakeholders also influence the project, such as top management. An example is that a low supported project by top management has a larger chance of failure (Cockburn & Highsmith, 2001). Top management support is found to be helpful in the software development processes when fulfilling certain roles. These roles are defined as: approving the return on investment (ROI) calculations for software projects, providing funding for software development projects, reviewing milestone, cost, and risk status reports, assigning key executives to oversight, governance, and project director roles and determining if overruns or delays have reduced the ROI below corporate targets (Jones, 2010). It should be noted that without proper reporting and therefore reporting disinformation projects might still fail.

Depending on the type of project, the involvement of a user increases the motivation of the development team and thereby increasing the success of the project. The involvement of users are found to be of importance in other studies (Chow & Cao, 2008; The Standish Group, 2009).

2.4.2 Development process

Experiences have shown that communicating requirements from the client to an end product did not meet the expectations of the clients. This plays a large role in building the right thing. In the order of building the thing right practices are introduced to improve the development processes. Agile practices used by the teams are there to structure the development process and support the teams. It is not known which practices are necessary for a project to be successful.

User participation in some studies was found to be of positive influence to the outcome. However, in many studies it was inconclusive, in which, most positive relations were found in empirical studies. Development managers, systems developers, users, and user managers perceived user participation as important to system success. A lack of user participation was also found to be a risk factor to system failure or abandonment. In agile

projects, user participation is found to be of importance. On the other hand a high participation of a user (>60%) is found to be not desirable by both developing team and client (Subramanyam, Weisstein, & Krishnan, 2010). In the extension to user participation, training for users are also found to be of importance to the success of a system. The acceptance or rejection of the system will be determined by the training. When user training is introduced early in the development process, instead of after installation, it may contribute to the development.

2.4.3 Project content

From a project point of view, the level of resources can determine the level of success of a project. These resources can consist of: amount of money, people and time for development. Limited resources could be perceived as low support and commitment of senior management to the project, which can demotivate the members of the project.

Furthermore, an achievable scope and clear project goals or objectives is found to be important by a number of studies. A large project scope, underestimating the scope of a project, changing scope or objectives, unclear goals or objectives, lack of agreement on goals or objectives among interested parties (e.g. management, information systems staff, users), or elusive goals that emerge and change as the project proceeds are suggested to result in less successful projects. On the other hand agile is suggests to let the scope vary such that it does not affect the cost, quality or the schedule of a project.

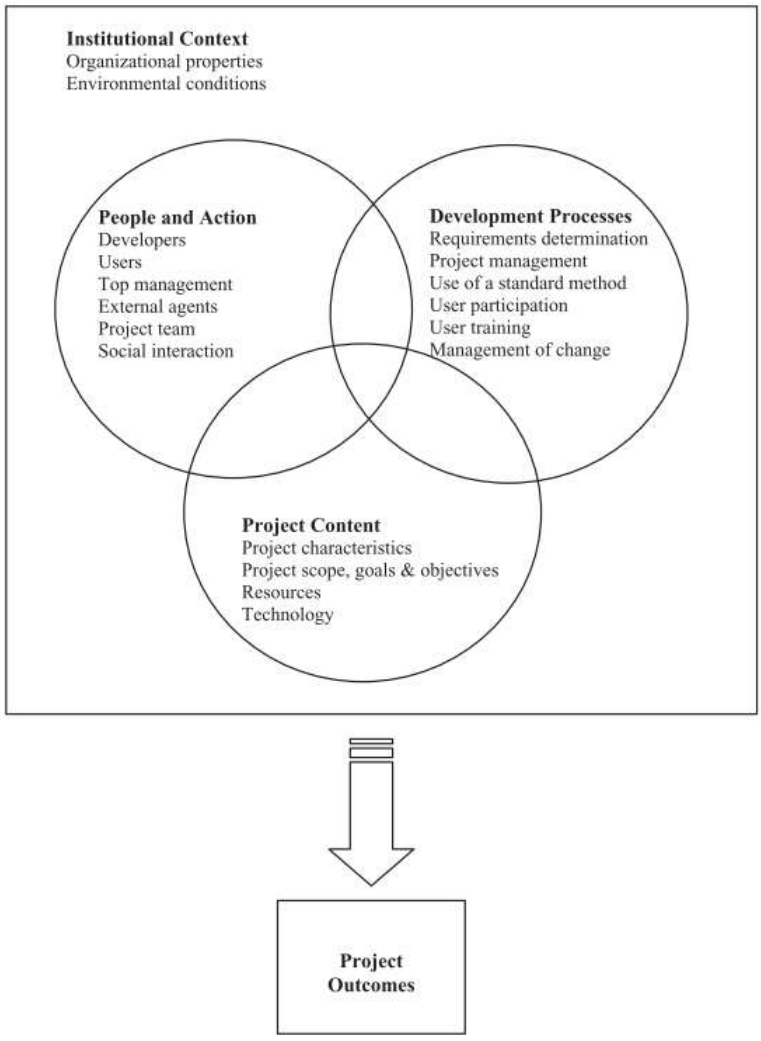


Figure 6. Project outcome framework McLeod and MacDonell (2011)

This current study investigates whether the model of McLeod and MacDonell (2011) can be used as a measurement system of agile projects for portfolio management. A reason to use this model is that the framework includes all factors that can influence the outcome of projects, also within larger organisations. Furthermore, it is currently the most advanced research done that describes these influences.

3 METHODOLOGY

3.1 RESEARCH DESIGN

This study will be based on the existing theory of McLeod and MacDonell (2011). It will try to assess quantitative data based on the qualitative model they describe. Järvinen (2001) defines constructive research as typically involving the building of a new innovation based on existing (research) knowledge and new technical or organisational advancements (Salo, 2006). This study can therefore be classified as constructive research. According to Järvinen (2001), it is possible to accept a prototype or a plan instead of a full product when doing constructive research. The research strategy of this study is that of a multiple case study (Yin, 2003).

3.2 RESEARCH METHOD

In section 2 we have found which agile measurements are compatible with other software development methods. Also, measurement methods were found to measure communication and motivation to be integrated in the framework of McLeod and MacDonell (2011) describing project outcomes. The study will be based on this framework that function as a guide for finding the project outcomes in practice. Project outcome is usually defined as success or failure, which may seem as a two-dimensional result, but is actually multi-dimensional. As McLeod and MacDonell (2011) mention in their article, the framework is a reference, which can be used in depth in order to generate more detail of the outcome of a certain project. To find the factors influencing the outcomes several questionnaires were conducted and interviews were done on projects in multiple organisations. To gather organisations to participate in the study a mailing list was used containing contacts in 800 organisations. Of this list 6 individuals responded willing to help with the study. However, the organisations where the individuals were part of did not fit the criteria. The main part of the organisations that responded was not large enough and the other part did not create software but only have a directing role. The inclusion criteria for the selection of organisations for this study were:

- Organisations developing software for internal or external use

- Organisations developing software in an agile manner
- Organisations consisting of more than 100 employees
- Organisations having the need of portfolio/program management, thus having to monitor multiple projects

In order to link the factors that were found by means of questionnaires and interviews, literature was prompted for the setup of questions and for the detail of the questions.

To find the association between the model and agile projects the first step was gathering data to prove associations. This was done by means of two questionnaires and two interviews. The questionnaires are found in Appendix D: Questionnaires and interviews. The connection between the framework and the properties of the domains are further elaborated here. If these associations exist, it could be used for a measurement system.

3.2.1 Questionnaire design

In the tables below the sources of the questionnaires are given that will give insight in the People & Action and Development Process dimension. The categories that lack a source consisted of questions that were not derived from existing questionnaires. Later in this section the construct of the questionnaires will be further explained.

Questionnaire 1 (Sources)

Category questions	Source(s)
General information	-
Goal commitment	(So, 2010)
Social support	(So, 2010)
Open communication	(So, 2010)
Adaptation	(So, 2010)
Coordination capability	(So, 2010)
Knowledge growth	(So, 2010)
Team performance	(So, 2010)
Additional questions	-

Table 4. Sources of questions of the first questionnaire (Q1)

Questionnaire 2 (Sources)

Category questions	Source(s)
General information	-
Requirements determination	(Jones, 2010)
Agile development	(Williams, 2012)
Software engineering & programming	(Jones, 2010)
User training	(Nelson, 1987)
SPICE practices	(SPICE, 1995)
Requirements determination construction	(Felici, 2004)
User participation	(S. C. Misra, Kumar, & Kumar, 2009; Subramanyam et al., 2010)
Project management	(Cerpa & Verner, 2009; Verner, Cox, Bleistein, & Cerpa, 2007; Verner & Evanco, 2005)
Senior management	-
Project and result rating	-
Additional questions	-

Table 5. Sources of questions of the second questionnaire (Q2)

3.2.1.1 People and Action

As mentioned in section 2.3.1, motivation was covered by questions measuring the self-reflection of a team. These questions formed a questionnaire directed to the development team and consisted of a few topics: Goal commitment, social support, open communication, adaptation, coordination capability, knowledge growth and team performance (So & Scholl, 2009). The survey was not used for the same purpose as in the original study.

The commitment of top management to a certain project affects project outcomes. Support and commitment are different in the type of supporting. A famous metaphor of involvement and commitment applies here: in the process of making a bacon-and-egg breakfast, the chicken is involved and the pig committed. Although involvement and support are different concepts, both do not show a level of tightness to the project. Top management must play a role in the project from initiation through implementation. If not,

it will be a risk overshadowing other risks (Keil, Cule, Lyytinen, & Schmidt, 1998). In this case the subject senior management reflected on the involvement, support and commitment of portfolio management in the project. Also the alignment and resourcing were covered. This was done using interview 1 and questionnaire 2. Other factors were not included in this study due to the limitations that are described in section **Error! Reference source not found.**.

3.2.1.2 Development Processes

To determine the requirements definition process, a part of a questionnaire was used covering how the requirements were setup and which software development methods were used during the process.

Top / Senior management processes were covered by an interview (I1) with a portfolio manager or a person with an equal role.

The project management property of the domain development processes was split up into practices that prove to be most practiced in agile software development (Williams, 2012) and best practices in software engineering in general (Jones, 2010). From the lists of both sources a fragment was extracted to form the list that was used in the questionnaire. The questions in the survey about project management were constructed using existing studies (Jones, 2010; Konrad, Paulk, & Graydon, 1995; Williams, 2012).

The user participation questions were based on a study where user participation in software development projects were analysed. In an average project, the user(s) generally participated in the following activities: project scoping and prioritization of requirements, responding and providing inputs to product prototypes created by development teams (these prototypes were primarily addressing user interface issues and communication functionality) and participating in design meetings and providing inputs on product features as well as design aspects that were important to the user group (Subramanyam et al., 2010). Furthermore, a survey question concerning closeness of collaboration between users and developers was used (S. C. Misra et al., 2009).

In order to measure user training a list created in 1987 was used that consisted of the following training techniques: tutorial, courses/seminars, computer-aided instruction,

interactive training manual, resident expert, help component and external (training) (Nelson, 1987). .

For the domain Development Processes the Baseline Practices Guide (Konrad et al., 1995) of the SPICE assessment has been advised. The Baseline Practices Guide defines, at a high level, the goals and fundamental activities that are essential to good software engineering (Paulk et al., 1995). The agile principles and best practices were adopted measures in (agile) software development processes. The agile practices were gathered from an article containing essential agile practices according to agile practitioners (Williams, 2012). The list of best practices for the software engineering section in the questionnaire were found by Jones (2010), which was constructed from a dataset containing 13000 projects and 600 organisations.

The similarity of a set of questions was expressed by the reliability analysis (Cronbach's alpha). The questions in the topic "Requirement determination construction", "User participation", "Project Management" and "Senior management" (Q2) were tested for internal consistency, because these have not been validated to measure the same concepts. For this, the alpha needed to be greater than 0.7. The questions within the category "Requirements determination construction" had an α of 0.87, "User participation" had an α of 0.91, "Project management" had an α of 0.72 and "Senior Management" had an α of 0.95. These are also found in Q2 of Appendix D: Questionnaires and interviews.

	Cronbach's Alpha	# of items
Requirements determination construction	0.87	6
User participation	0.91	5
Project management	0.72	6
Senior management	0.95	3

Table 6. Reliability analysis

Some of the questions are negated, therefore for the analysis the values of these variables need to be negated as well. This is the case for the questions within Q1: all questions in the category "Open communication", all questions in the category

"Coordination capabilities" and question 4 of "Goal commitment". In Q2, the variable that needs to be negated is: question 6 of "Project management".

3.2.2 Interview design

For the Project Content dimension the properties that were covered are the resources and characteristics of the project. These consisted of artefacts of the project respectively: financial resources, development time, human resources and size, complexity. The development time was expressed in man hours. The human resource factor was defined by the scope and alignment of goals, as described by the model of McLeod and MacDonell (2011). The human resource factor was measured with an interview with the person responsible of the program or portfolio of the organisation.

3.2.3 Chain of evidence

The data was gathered using three methods of data gathering, which were: interview, questionnaire and artefact. The questionnaire design is found in 3.2.1 and the interview design in 0. The artefacts were gathered via the interviews. By gathering the data using these different methods relying on different sources the results were triangulated. The definition of triangulation is:

"attempt to map out, or explain more fully, the richness and complexity of human behavior by studying it from more than one standpoint." (Cohen, Manion, & Morrison, 2007, p 254)

Figure 7 is the graphical representation of how the framework was triangulated. There should be noted that interview I2 mostly consisted of questions that require the interviewee to retrieve the data from the project details. More specifically, documents containing project details were asked. The rest of the questions in the interview referred to the process of the project.

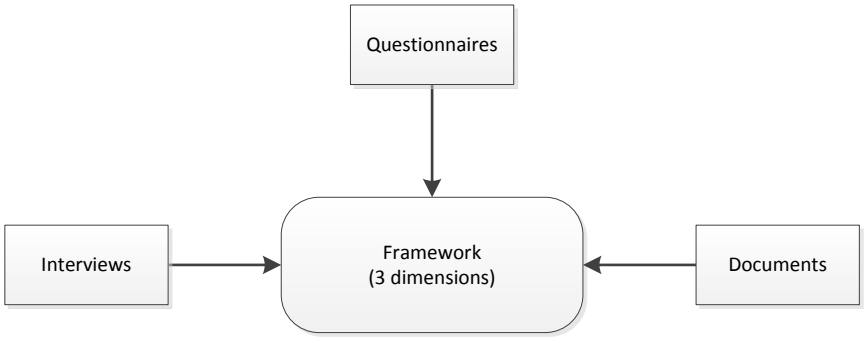


Figure 7. Triangulation with the unit of analysis

A general picture of the questionnaires and interviews can be found in Figure 8. Different from the model found in the paper of McLeod and MacDonell (2011) was the management process. Although also indirectly found in the framework ("top management"), it grasped how the project was linked to and supported by the interviewee of Interview I1. Furthermore, the interview was setup to find information required to be reported to program or portfolio management. From the three sources also the three dimensions of the framework were covered. Missing in this figure is the *Institutional Context* that is described in the study of McLeod and MacDonell (2011). The reason to leave it out was the scope of the study.

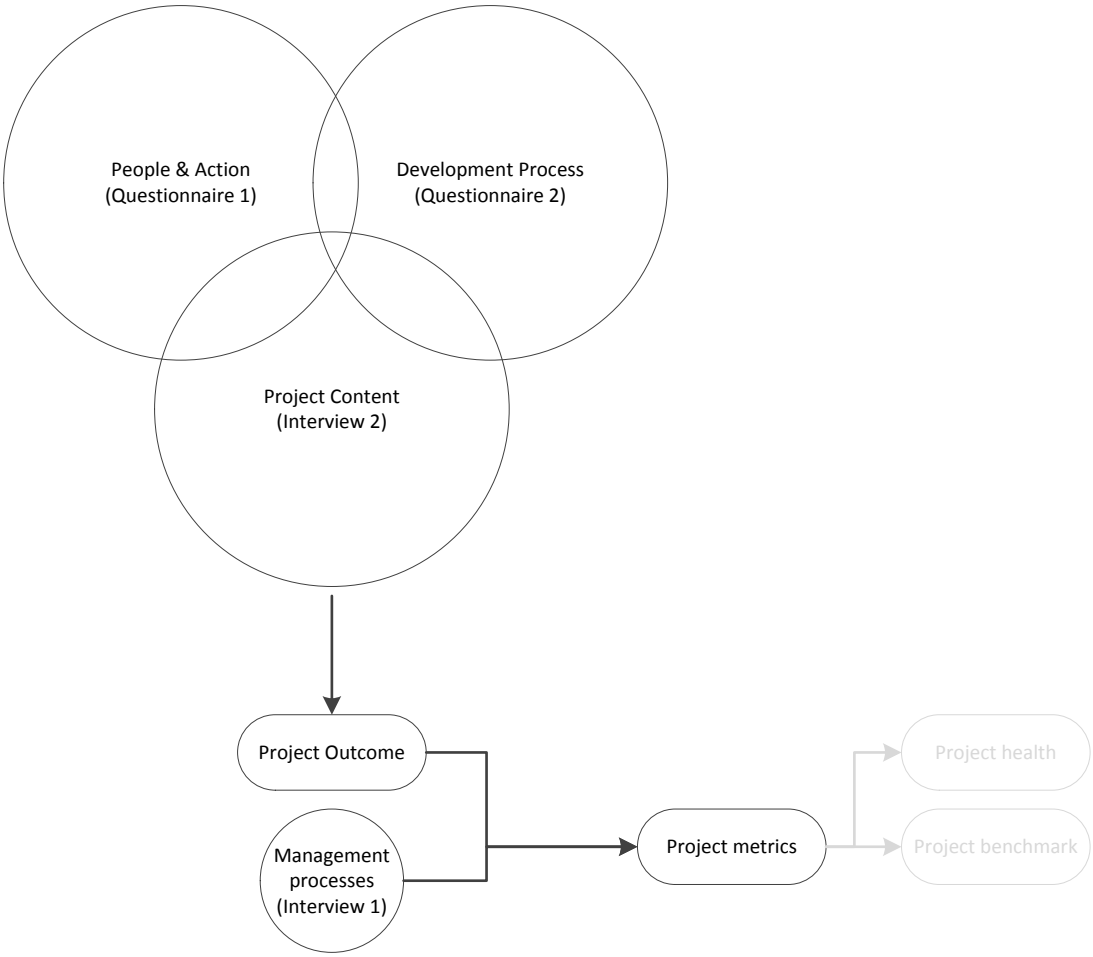


Figure 8. Connection of data gathering methods and framework

The data was gathered by means of triangulation; one source of data being the questionnaires, the second source of data being the interviews and the last source the documentation of the project. One of the purposes of the interviews was to gain insight on the why and how of the processes being followed. Combined they gave the knowledge of how the project outcome had come to place from different perspectives.

The interviews and questionnaires are combined in a case study database. Here the taped and transcribed interviews are stored per organisation and labelled with the role of the interviewee. The results of the questionnaires are stored separately in a single file each instance labelled with organisation and project number.

Several organisations were involved in the study in order to have at least a significant amount of responses on both questionnaires. Furthermore, these organisations followed a portfolio/program that steered the projects.

3.3 VALIDITY

According to Yin (2003), the quality of the design of a case study can be best evaluated by four tests, being: Construct validity, internal validity, external validity and reliability.

To comply to the test of construct validity, Yin (2003) state that the investigator needs to select the specific types of changes that are to be studied (and relate them to the original objectives of the study) and demonstrate that the selected measures of these changes do indeed reflect the specific types of change that have been selected.

Table 7 shows the sections where each of the types of validity are described in more detail.

Test	Case study tactic	Explanation
Construct validity	Use multiple sources of evidence (Interviews, questionnaires, documents)	Section 3.5
	Establish chain of evidence (i.e. interviews taped, transcribed)	Section 3.2
	Have key informants review draft case study report	-
Internal validity	Pattern-matching (i.e. statistical analysis)	Section 3.6
	Explanation-building (i.e. relate analysis to literature and results)	Section 3.6
	Logic models (i.e. Figure 8)	Section 3.2
External validity	Use replication logic in multiple-case studies (taken 5 organisations in this study)	Section 3.5
Reliability	Develop case study database (i.e. storing results of interviews per organisation)	Section 3.5
	Use case study protocol (i.e. semi structured interviews, questionnaires on paper)	Section 3.2, 3.3, 3.5, 3.6,

Table 7. Case study validity (Yin, 2003)

3.4 PLANNING

The workflow of this study is found in Figure 8. The start of the project was in the last week of November 2012 and ended in June 2013.

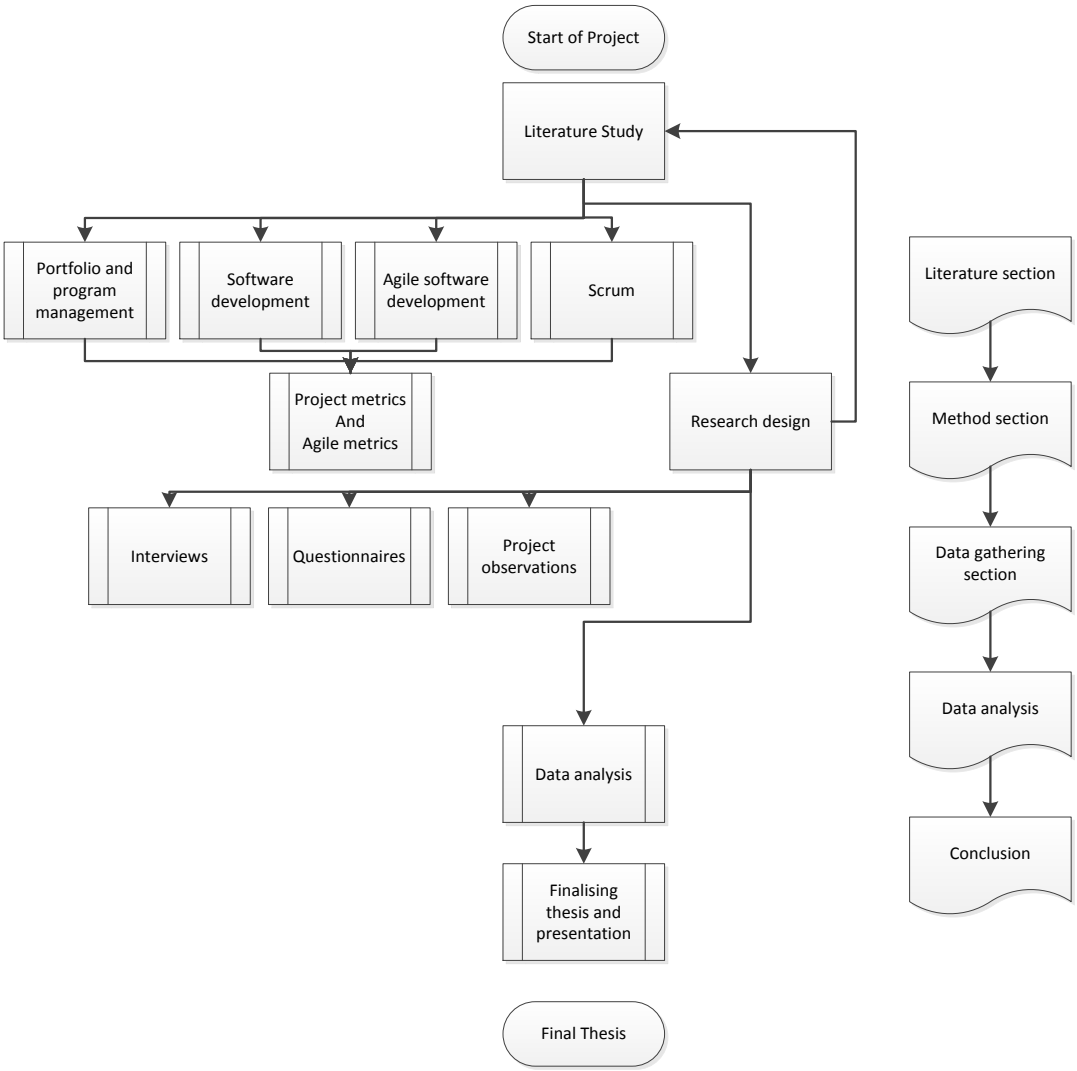


Figure 9. Planning of the study

3.5 DATA COLLECTION

3.5.1 Pre study

Prior to the data collection the second questionnaire Q2 has been tested if it was understandable and also timed to determine the duration for a participant to fill in the

questionnaire. The first questionnaire was validated, so did not need to be tested. Q2 was tested on two subjects, where one was part of the teams participating in the study and the other in a project that would not be included. Based on the outcome, questions were asked, such as "*Which questions did you find difficult to answer?*" and "*Which questions were unclear to you?*". A follow up question was "*In what sense did you find it difficult to answer?*" to find out where improvements could be made. The questionnaire was adapted to the feedback gained from the sessions. It did not require changes to the phrasing of the questions, but elaboration of term descriptions were added to the questions.

3.5.2 Collection planning

The timeline found in Figure 10 shows when the interviews and questionnaires were measured in a given time of a certain agile project. The general idea for this study was to measure the questionnaires, both Q1 and Q2, at one time in a project. This moment was shortly after a release or a sprint.

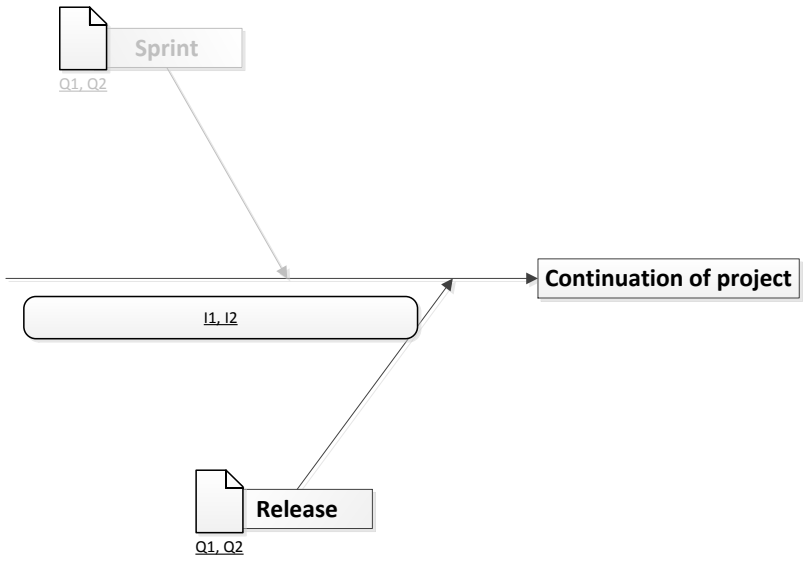


Figure 10. Timeline of a project and when measurements are ideally to take place

More specifically, the questionnaires regarding the processes were gathered after each release (Q2). The reason to measure at the end of a release is because of the closure of

a project, where data such as budget overrun could be taken into analysis. For some projects a distinction was made due to the time constraint of this study. Here the sprint was taken as a measurement point with the reason that a sprint contains a potentially shippable product. An interview with the portfolio holder (I1) and the gathering of the documentation regarding the project information (I2) to the last release occurred at the same frequency.

All information was stored in one database where the sources were stored per project. The surveys were stored separately in one database.

In order to include a significant amount of respondents to be able to say something about the association between the factors and project outcome, several organisations were asked to participate in the study. These organisations were asked to participate by using a mailing list with contacts within an organisation. Also contacts of the researcher were asked to participate. The goal was to have as many organisations contributing to the study as possible, with a minimum of 50 respondents for the questionnaires that were working in teams and divided over several organisations.

3.6 DATA ANALYSIS

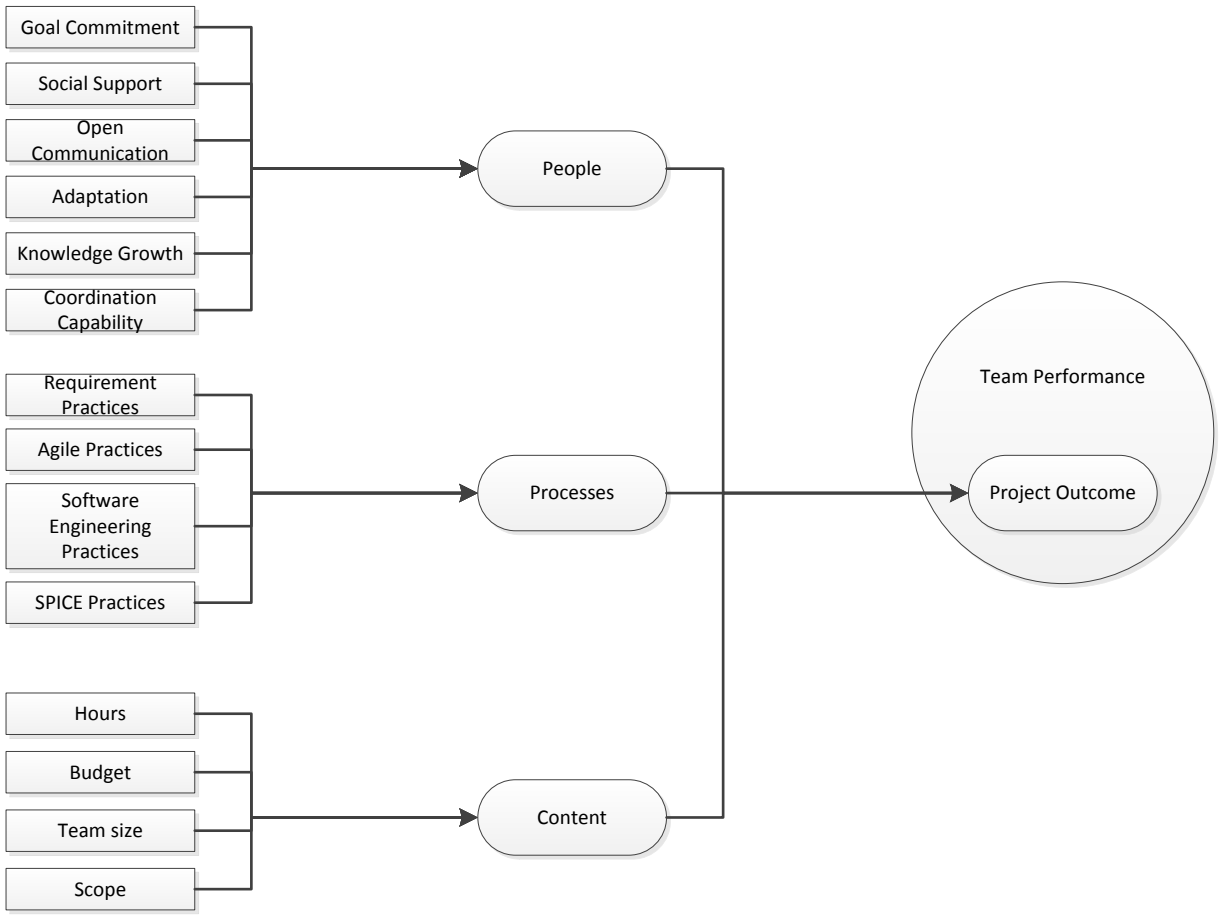
The data was analysed using SPSS 20. Project outcome was measured with the question *"How do you rate the project in general on a scale from 1 to 10, where 10 is excellent?"* and was used in the statistical analysis as the dependent variable. As product outcome is a subset of project outcome the question measuring product outcome is measured by: *"How do you reate the result on a scale from 1 to 10, where 10 is excellent?"*. A correlation is found between the question measuring project outcome and product outcome to have a correlation coefficient of 0.601, which means the two variables are moderately correlated (Calkins, 2005). Project outcome was considered a continuous variable and therefore linear regression analysis was used. The results of the questionnaire were used to find an association between the (independent) variables and this dependent variable. The independent variables were hypothesized to contribute to the dependent variable, e.g. motivation should increase project outcome.

Project outcome was used to find the association with the aspects in the development process, such as the use of specific requirements practices, agile practices and software engineering practices. Also, the questions covering the teamwork aspects were analysed for associations. As the result and determinants (i.e. deliver on time, on budget, high quality, etc.) is a subset of the project outcome, these will also be associated with the used practices. Regression analysis was also used on each of the significant related determinants and the factors.

Another example to analyse is: The amount of resources is an important influential factor, because a lack will demotivate the developers (McLeod & MacDonell, 2011). Interviews with the program or portfolio managers explained the management process parallel to the development process and how it affected certain aspects in the project. From the interview challenges for the processes were derived and compared with the other interviews to arrive at a conclusion which measurements are required for program or portfolio management.

The practices that were not directly influencing the outcome of a project were being associated with the determinants that were directly associated and were found to be contributing to the outcome. The increasing number of a factor on a determinant does not necessarily mean an increase in project outcome. However, it does increase the chance on a higher project outcome.

Data analysis plan is as follows:



4 RESULTS

This section shows the results found after data collection and analysis. The results are divided into sections organised by organisation and method that was used to gather the data. There are several organisations where the data will be gathered from. These organisations and projects will be kept anonymous.

In this study, 5 organisations participated all within different industries. Both interviews and questionnaires were taken in these organisations. In total there were 48 respondents of the questionnaire distributed over 9 projects. Next to the questionnaires, 11 interviews were held with portfolio/program managers and project manager.

4.1 CASE DESCRIPTIONS

4.1.1 Organisation 1

The organisation where this study was initiated place is a software vendor that produces medical related software to medical organisations. Due to a recent takeover the organisation needs to adapt to regulations and changes undergoing in the organisation. Furthermore, the organisation exists around 40 years and switched from the waterfall method (and RUP) to the agile method a few years ago in order to cope with the changing environment. Due to the long duration the employees have worked with the waterfall method, mind sets need to be changed in order for agile to be fully exploitable. With help of a consultancy firm that coaches scrum, divisions within the organisation are getting knowledge of how the new way of working has to be applied and how this affects the outcome of the projects.

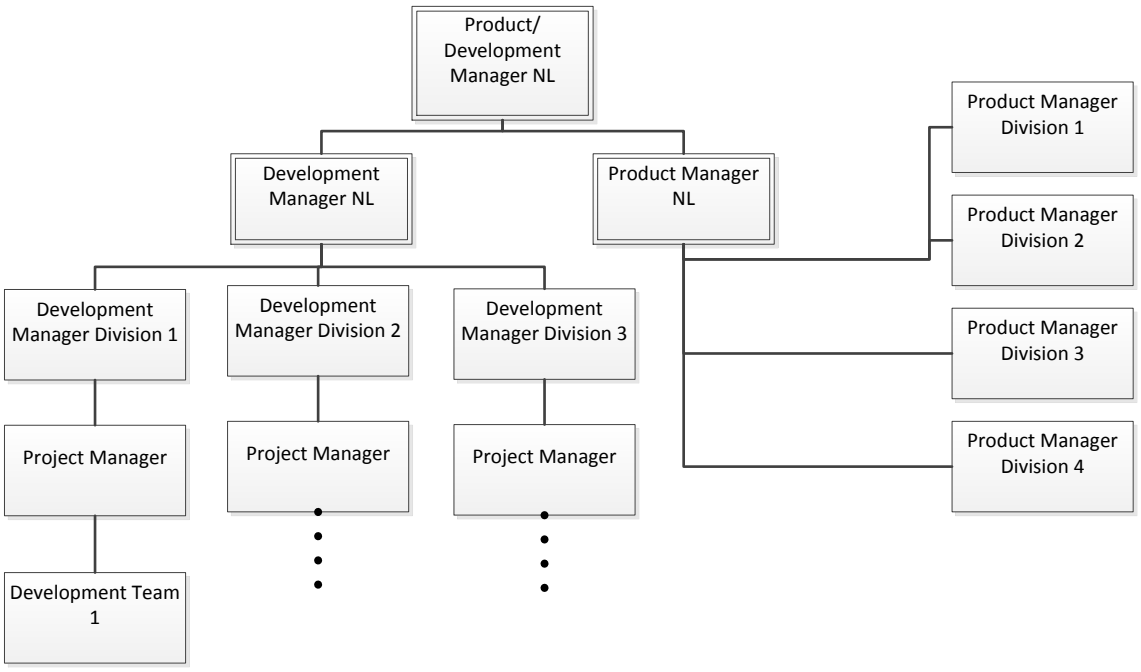


Figure 11. Organisational Structure of the development division

4.1.2 Organisation 2

Organisation 2 is an organisation with its general task is to pay and receive individuals according to law or law changes. The latter is where the development department is currently working on. Changes made in law by the government should be applied to the process that is found in the administration system. A process is broken up at a point where the modification is to be made. In a simple form the module can be a parallel path of a process. The amount of law changes and the size and drag of the changes translate into a complexity several development teams can work on. As the organisation is found in the public sector the client is the government. It requires the organisation to rectify the duration and expenses of the project.

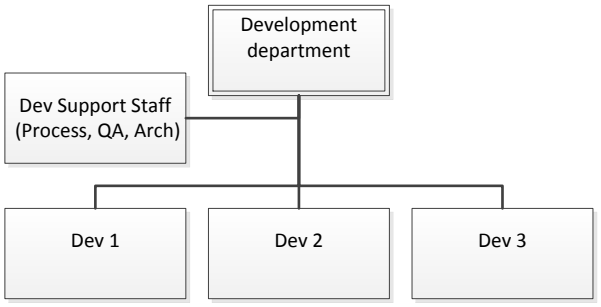


Figure 12. Development department structure of Organisation 2

4.1.3 Organisation 3

Organisation 3 is the same company as Organisation 1, but situated in a different location and operating in different fields. The two organisations therefore currently work independently. This organisation has several divisions where one project that does not only develop software, but also involves the hardware.

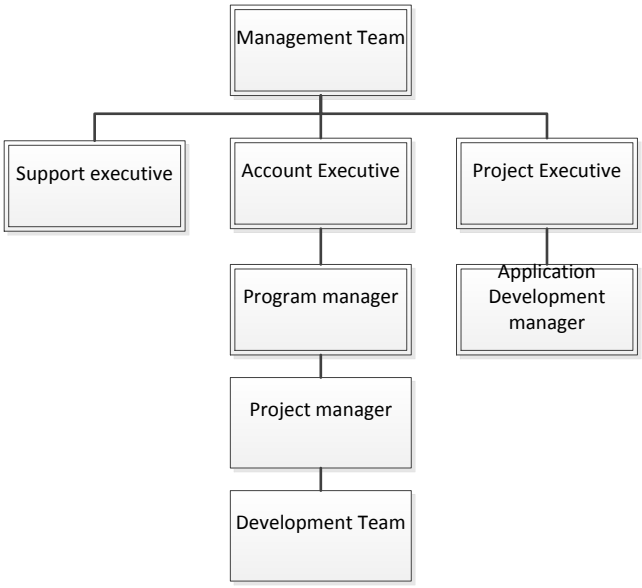


Figure 13. Part of the structure of organisation 3

4.1.4 Organisation 4

Organisation 4 is active in the e-commerce industry. Their core product is their website, which consists of multiple categories. These categories need to be maintained and updated. Also general support and linking of the structure is required. Development teams are usually fixed to certain categories.

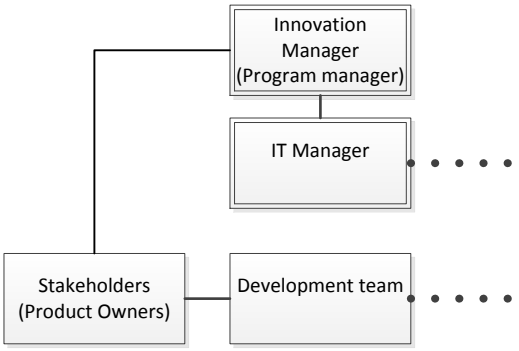


Figure 14. Reporting structure of organisation 4

4.1.5 Organisation 5

Organisation 5 is an organisation active in the telecom industry. Furthermore, it is also one of organisations with the larger market share. The majority of the development teams develop the software using the waterfall method. Currently the organisation has one scrum team. Furthermore, it is looking into expanding the method to other the development teams with one of the reasons of the ability to market faster.

4.2 SURVEY

4.2.1.1 Respondents

The respondents of the questionnaires are members of the development team. Within the team there are different roles. Figure 15 shows which roles participated.

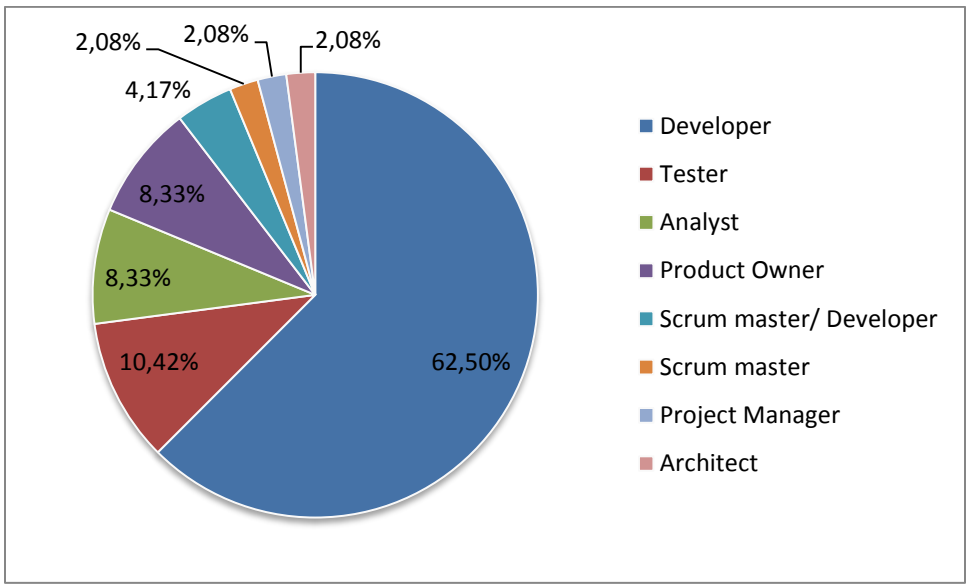


Figure 15. Roles of the respondents

Of the number of respondents 62.5% indicated that their role is a developer. All roles such as client developer, server developer and senior developers are covered by this. Added to this percentage can be the 8.33% respondents that also have the role of scrum master next to being a developer. Adding the percentage the total number of developers is: 70.83%.

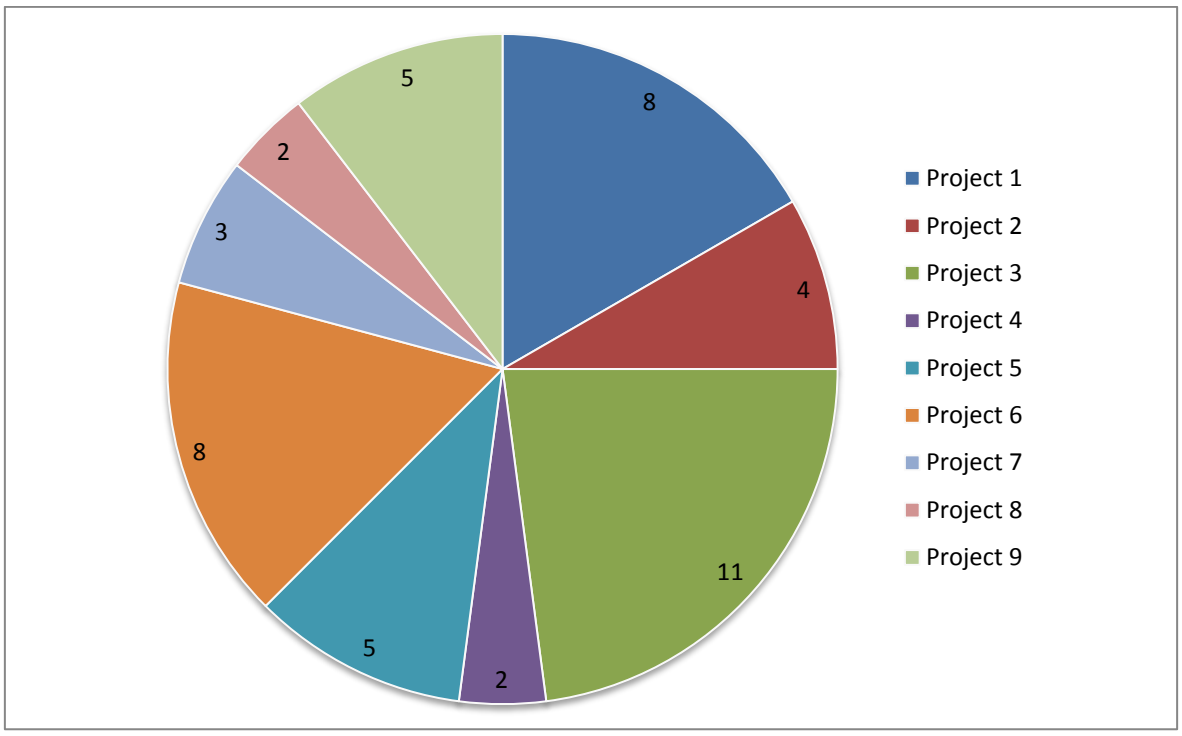


Figure 16. Participating projects

Figure 16 shows the number of responses per project. The size per project differs, where 6 projects had 4 or more respondents and the rest less than 4 respondents. The teams where the respondents of the questionnaires were in were distributed over 9 projects.

The details of the projects that were revealed during the interviews or documents can be found in Table 8. It shows the projects that participated in the study and the characteristics of the project the team was working on.

Project # / characteristics	1	2	3	4	5	6	7	8	9
Organisation	1	1	1	1	2	2	3	4	5
#teams	3	1	2	2	2	N/A	1	1	1
#team members	8;3;2	10	6;8	3;3	4;5	N/A	9	7	N/A
Project length (months)	12	18	14			N/A			
Sprint length (weeks)	2	2	2	2	2	2	4/5/6	1/2	3
Hours budgeted	6000	10000	14000	N/A	N/A	N/A	N/A	N/A	N/A
Hours estimated	42%	106%	122%	N/A	N/A	N/A	N/A	N/A	N/A
Rounded hours used	109%	115%	81%	5000	N/A	N/A	N/A	N/A	N/A
Rounded costs	330k	650k	540k	240k	N/A	N/A	N/A	N/A	N/A
Avg. project rating	6.38 σ = 0.46	7 σ = 0.71	6.25 σ = 0.37	6 σ = 1.00	5.4 σ = 0.68	7 σ = 0.31	6 σ = 0.58	7.5 σ = 0.50	9 σ = 0.00

Table 8. Projects and their characteristics

Figure 17 shows the average motivation of each project. Project 2 is found to have the least motivation with an average of 3.25, where project 9 has the highest motivation 4.25.

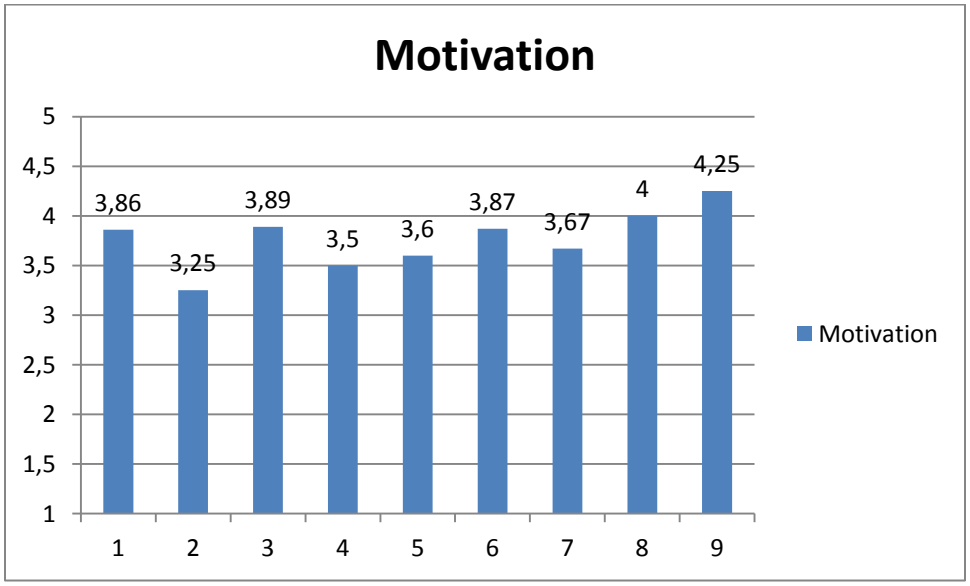


Figure 17. Average motivation of team members in the projects

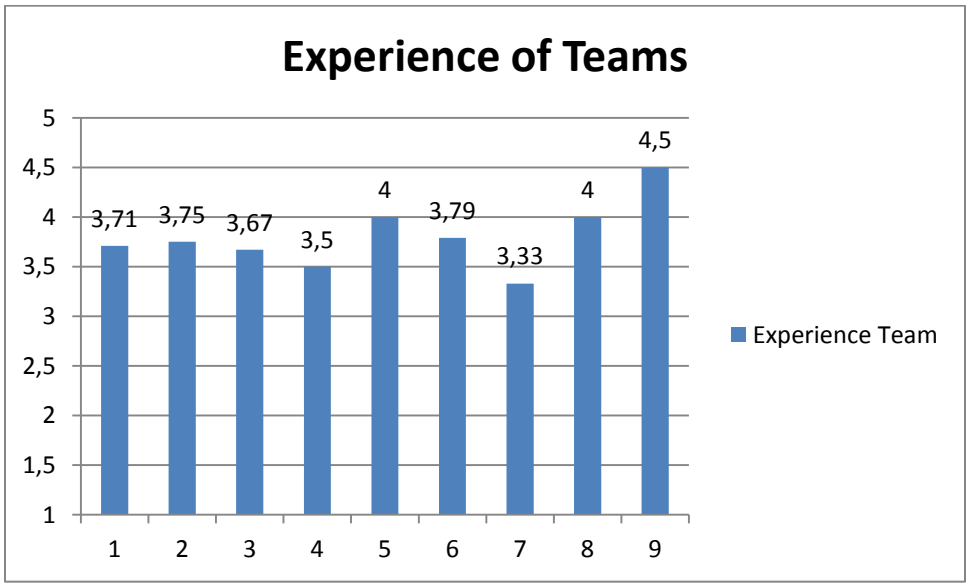


Figure 18. Average experience in the teams

The average experience of the teams can be found in Figure 18. The team of project 9 has the highest average of experience with a number of 4.5. Project 7 has the lowest with a number of 3.33.

4.2.1.2 Teamwork

For each project a spider graph has been plotted to see measure the performance of the teams within a projects for the categories: coordination, goal commitment, social support, open communication, adaptation and knowledge growth.

	Coordination		Goal commitment		Social support		Open communication		Adaptation		Knowledge Growth	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ	Mean	σ
Project 1	4.85	0.37	4.71	0.77	5.60	0.89	5.33	0.94	5.00	0.69	4.75	0.83
Project 2	4.46	0.50	4.85	0.43	5.65	1.00	5.75	0.50	4.94	0.31	4.50	0.19
Project 3	4.76	0.59	5.13	0.68	5.62	0.50	5.07	1.07	4.50	0.78	4.33	0.49
Project 4	4.67	0.71	4.67	0.94	5.90	1.56	4.70	0.42	4.00	0.00	5.25	1.53
Project 5	3.80	1.38	5.20	0.46	**6.12**	0.30	5.20	0.84	4.70	0.69	4.43	0.51
Project 6	4.88	0,55	5.47	0.57	5.93	0.66	5.33	0.57	4.66	0.80	5.00	0.46
Project 7	4.94	0.42	5.11	1.42	5.73	0.31	5.13	0.70	2.75	1.52	4.67	1.04
Project 8	5.08	0.35	5.83	0.00	5.20	0.85	**6.00**	0.28	5.13	0.18	4.25	0.59
Project 9	**5.43**	0.95	**5,88**	0.67	5.65	0.55	5.85	1.18	**5.25**	0.61	**5.33**	0.72

Table 9. Teamwork measured on a scale of 1 to 7

In the figures that are found in Appendix E: Spider graphs of teams, show the teamwork of each project is plotted. Seen in Table 9, project 5 scores the least in the category Coordination. Project 9 scores the highest in the categories: coordination (5.43), goal commitment (5.88), adaptation (5.25) and knowledge growth (5.33). In the category social support project 5 scores the highest with 6.12 and for open communication project 8 scores the highest with 6.00. All projects score a higher than the average score (4) for all categories except project 5 for coordination (3.80) and project 7 for adaptation (2.75).

4.2.1.3 Requirements engineering practices

In the category of requirements best practices the results found in Table 10 came out of the survey.

Requirements Engineering Practice	Answer	Percent			
Joint client/vendor change control board	Yes No, but should have No, not necessary I don't know	33.3% 12.5% 25.0% 14.6%	**Quality function deployment**	Yes No, but should have No, not necessary I don't know	10.4% 16.4% 29.2% 31.3%
Domain experts for changes to specific features	Yes No, but should have No, not necessary I don't know	64.6% 12.5% 10.4% 0%	**Security analysis and vulnerability prevention**	Yes No, but should have No, not necessary I don't know	31.3% 16.7% 20.8% 18.8%
Requirements traceability is present	Yes No, but should have No, not necessary I don't know	37.5% 29.2% 12.5% 8.3%	**Prototypes for key features of new applications**	Yes No, but should have No, not necessary I don't know	52.1% 18.8% 12.5% 4.2%
Multiple releases of requirements changes	Yes No, but should have No, not necessary I don't know	41.7% 10.4% 25.0% 6.3%	**Full-time user involvement for agile projects**	Yes No, but should have No, not necessary I don't know	39.6% 25.0% 20.8% 2.1%
Utilization of automated requirements analysis tools	Yes No, but should have No, not necessary I don't know	8.3% 16.7% 27.5% 22.9%	**Mining legacy applications**	Yes No, but should have No, not necessary I don't know	43.8% 2.1% 25.0% 16.7%
Careful analysis of the features of packages	Yes No, but should have No, not necessary I don't know	47.9% 4,2% 10.4% 22.9%	**Clear and understandable requirements**	Yes No, but should have No, not necessary I don't know	54.2% 20.8% 10.4% 2.1%
Joint application design for initial requirements gathering	Yes No, but should have No, not necessary I don't know	54.2% 6.3% 8.3% 18.8%	**Formal requirement inspections with both users and vendors**	Yes No, but should have No, not necessary I don't know	25.0% 25.0% 18.8% 16.7%

Table 10. Requirements engineering best practices

The three highest practices that are practiced by the respondents are: domain experts for changes to specific features (64.6%), joint application design for initial requirements gathering (54.2%) and clear and understandable requirements (54.2%). Furthermore, the respondents that are not practicing a certain requirements engineering best practice have divided opinions whether it is necessary to be taken into the process.

4.2.1.4 Agile practices

Of the respondents the following list, of agile practices that are used in the organisations the following practices, is found that is used/not used in the projects:

Agile Practice	Answer	Percent			
Continuous Integration	Yes No, but should have No, not necessary I don't know	58,3% 14,6% 12,5% 0%	**Stand up**	Yes No, but should have No, not necessary I don't know	66,7% 6,3% 14,6% 0%
Short Iterations (< 30 days)	Yes No, but should have No, not necessary I don't know	83,3% 2,1% 2,1% 0%	**Small teams (12 people or less)**	Yes No, but should have No, not necessary I don't know	79,2% 6,3% 0% 0%
Definition of done	Yes No, but should have No, not necessary I don't know	66,7% 18,8% 2,1% 0%	**Co-located team**	Yes No, but should have No, not necessary I don't know	60,4% 16,7% 8,3% 2,1%
Automated tests run with each build	Yes No, but should have No, not necessary I don't know	35,4% 39,6% 12,5% 0%	**Coding standard**	Yes No, but should have No, not necessary I don't know	52,1% 20,8% 0% 14,6%
Automated unit testing	Yes No, but should have No, not necessary I don't know	33,3% 39,6% 14,6% 0%	**Pair programming**	Yes No, but should have No, not necessary I don't know	39,6% 8,3% 33,3% 4,2%
Iteration reviews	Yes No, but should have No, not necessary I don't know	83,3% 4,2% 0% 0%	**Burndown charts**	Yes No, but should have No, not necessary I don't know	58,3% 16,7% 10,4% 0%
Whole multidisciplinary team with one goal	Yes No, but should have No, not necessary I don't know	70,8% 10,4% 0% 6,3%	**Requirements written as informal stories**	Yes No, but should have No, not necessary I don't know	66,7% 8,3% 12,5% 0%
Features in iteration are customer-visible/valued	Yes No, but should have No, not necessary I don't know	66,7% 10,4% 2,1% 6,3%	**Embracing changing requirements**	Yes No, but should have No, not necessary I don't know	47,9% 20,8% 12,5% 6,3%
Prioritized product backlog	Yes No, but should have No, not necessary I don't know	75% 10,4% 0% 2,1%	**Test-driven development acceptance testing**	Yes No, but should have No, not necessary I don't know	41,7% 31,3% 6,3% 8,3%
Retrospective	Yes No, but should have No, not necessary I don't know	72,9% 12,5% 2,1% 0%	**Code inspections**	Yes No, but should have No, not necessary I don't know	52,1% 27,1% 2,1% 6,3%
Collective ownership of code	Yes No, but should have No, not necessary I don't know	72,9% 12,5% 2,1% 0%	**Planning poker**	Yes No, but should have No, not necessary I don't know	60,4% 12,5% 6,3% 6,3%

Table 11. Applied agile practices

The least applied agile practices (< 50%) found in to be practiced are: Pair programming (39.6%), Automated test runs each build (35.4%), Automated unit testing (33.3%),

Embracing changing requirements (47.9%) and Test driven development acceptance testing (41.7%). Of these practices, 33.3% of the respondents found Pair programming not to be necessary to apply in their process. Automated unit testing and Automated test runs each builds are found to be useful in the process, respectively 39.6% and 39.6% have rated that the practices should be incorporated in the process. Also 31.5% found that test driven development acceptance testing should be found in their process. The practice embracing changing requirements have divided answers where 20.8% answered "no, but should have it" and 12.5% answered "no, not necessary".

4.2.1.5 Software engineering practices

Of the respondents the following list, of software engineering best practices that are used in the organisations the following practices, is found that is used/not used in the projects:

Software engineering Practice	Answer	Percent			
Early sizing and scope control	Yes No, but should have No, not necessary I don't know	58,3% 16.7% 4.2% 6.3%	**Version control**	Yes No, but should have No, not necessary I don't know	68.8% 10.4% 2.1% 4.2%
User involvement in software projects	Yes No, but should have No, not necessary I don't know	45.8% 20.8% 14.6% 4.2%	**Having clear and structured code**	Yes No, but should have No, not necessary I don't know	60.4% 18.8% 0% 6.3%
Selecting software methods, tools and practices	Yes No, but should have No, not necessary I don't know	31.3% 22.9% 14.6% 16.7%	**Formal code inspections of all modules**	Yes No, but should have No, not necessary I don't know	25.0% 31.3% 14.6% 14.6%
Having a software architecture and design	Yes No, but should have No, not necessary I don't know	68.8% 14.6% 0% 0%	**Clear and relevant comments in the source code**	Yes No, but should have No, not necessary I don't know	43.8% 22.9% 8.3% 10.4%
Selection of reusable code from certified sources	Yes No, but should have No, not necessary I don't know	39.6% 16.7% 20.8% 8.3%	**Re-inspection of code after significant changes or updates**	Yes No, but should have No, not necessary I don't know	33.3% 39.6% 2.1% 10.4%
Planning and including security topics in code	Yes No, but should have No, not necessary I don't know	12.5% 20.8% 29.2% 22.9%			

Table 12. Applied software engineering best practices

The most used software engineering practices (>50%) are having: a software architecture and design (68.8%), version control (68.8%), clear and structured code (60.4%), early sizing and scope control (58.3%).

4.2.1.6 SPICE practices

The total number of SPICE practices are 32. Table 13 contains the number of SPICE practices of a project. Respondents between projects answered to have an average of minimal 13 SPICE practices. The project the highest amount of SPICE practices is project 7 with a number of 21 practices.

	# SPICE practices	Std. dev
Project 1	17.43	6.35
Project 2	17.75	10.24
Project 3	14.75	11.65
Project 4	13	8.49
Project 5	15.75	6.08
Project 6	16.14	7.54
Project 7	21	5.29
Project 8	14.5	10.61
Project 9	18.5	2.12

Table 13. Number of SPICE practices per projects

4.2.1.7 Contribution to project outcome

The respondents of the questionnaire rated the project outcome on a scale of 1 – 10. Here the lowest number that was given was a 4 and the highest a 9. Figure 19 shows the expected distribution against the actual values, where it shows its linearity due to the dots being along the line. The average rating given for the rating of the project was 6.54 with a standard deviation of 1.3.

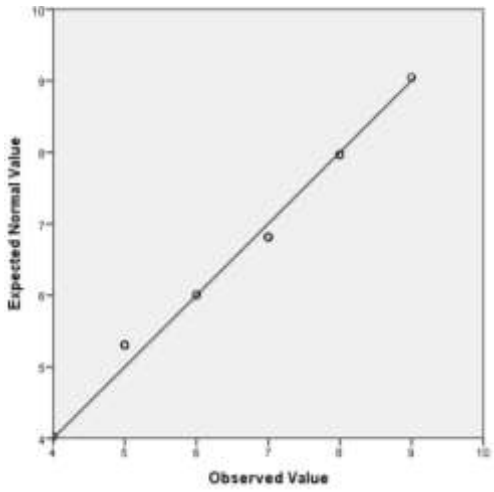


Figure 19. Q-Q Plot of the project outcome

The determinants of project outcome were rated on a scale of 1 to 7. The average of the determinants are found in Table 14.

Determinants (scale 1-7)	Mean	std deviation
Deliver on-time	4.36	1.64
Deliver on budget	4.21	1.78
Deliver high quality	5.09	1.15
Customer satisfaction	5.05	0.76
Lessons learned	4.94	1.16

Table 14. Descriptives of the determinants

Table 15 shows how the determinants: deliver on-time, deliver on budget, deliver high quality, customer satisfaction and lessons learned, contribute to project outcome when increasing one point on the scale. Respectively project outcome increases by: 0.351, 0.298, 0.423, 0.636 and 0.587 points.

Determinant	B	Std error	CI 95%
Deliver on-time	0.351**	0.116	0.116 – 0.586
Deliver on budget	0.298**	0.105	0.085 – 0.511
Deliver high quality	0.423***	0.163	0.093 – 0.754
Customer satisfaction	0.636***	0.256	0.117 – 1.155
Lessons learned	0.587*	0.156	0.271 – 0.902

*p-value ≤ 0.001
**p-value ≤ 0.01
***p-value ≤ 0.05

Table 15. Associations of the determinants with project outcome

Appendix F: Associations with determinants factors, contains the associations found between the determinants and factors.

Requirements practices	B	Std error	CI 95%
Clear and understandable requirements	1.133***	0.388	[0.347 – 1.92]
Agile practices	**B**	**Std error**	**CI 95%**
Coding Standards	1.133***	0.505	[0.105 – 2.162]
Embracing changing requirements	0.849***	0.409	[0.016 – 1.677]
Customer visible/valued features	1.306***	0.556	[0.178 – 2.435]
Planning poker	1.044***	0.492	[0.045 – 2.042]
People factors	**B**	**Std error**	**CI 95%**
Coordination	0.859**	0.254	[0.344 – 1.373]
Knowledge Growth	0.957*	0.264	[0.423 – 1.490]
Motivation	0.783*	0.228	[0.321 – 1.246]
Open Communication	0.615***	0.236	[0.137 – 1.093]
Senior management	0.285***	0.128	[0.026 – 0.545]
Team experience	0.821***	0.347	[0.118 – 1.524]
User participation	0.391**	0.130	[0.127 – 0.655]

*p-value ≤ 0.001
**p-value ≤ 0.01
***p-value ≤ 0.05

Table 16. Associations of factors with project outcome

Table 16 shows the associations found between the requirements, agile and software engineering practices and project outcome. Senior management is found to be associated to project outcome. The effect of senior management on project outcome is 0.285 points when this item increases by one point on the scale.

From Table 16 of the following figure can be derived:

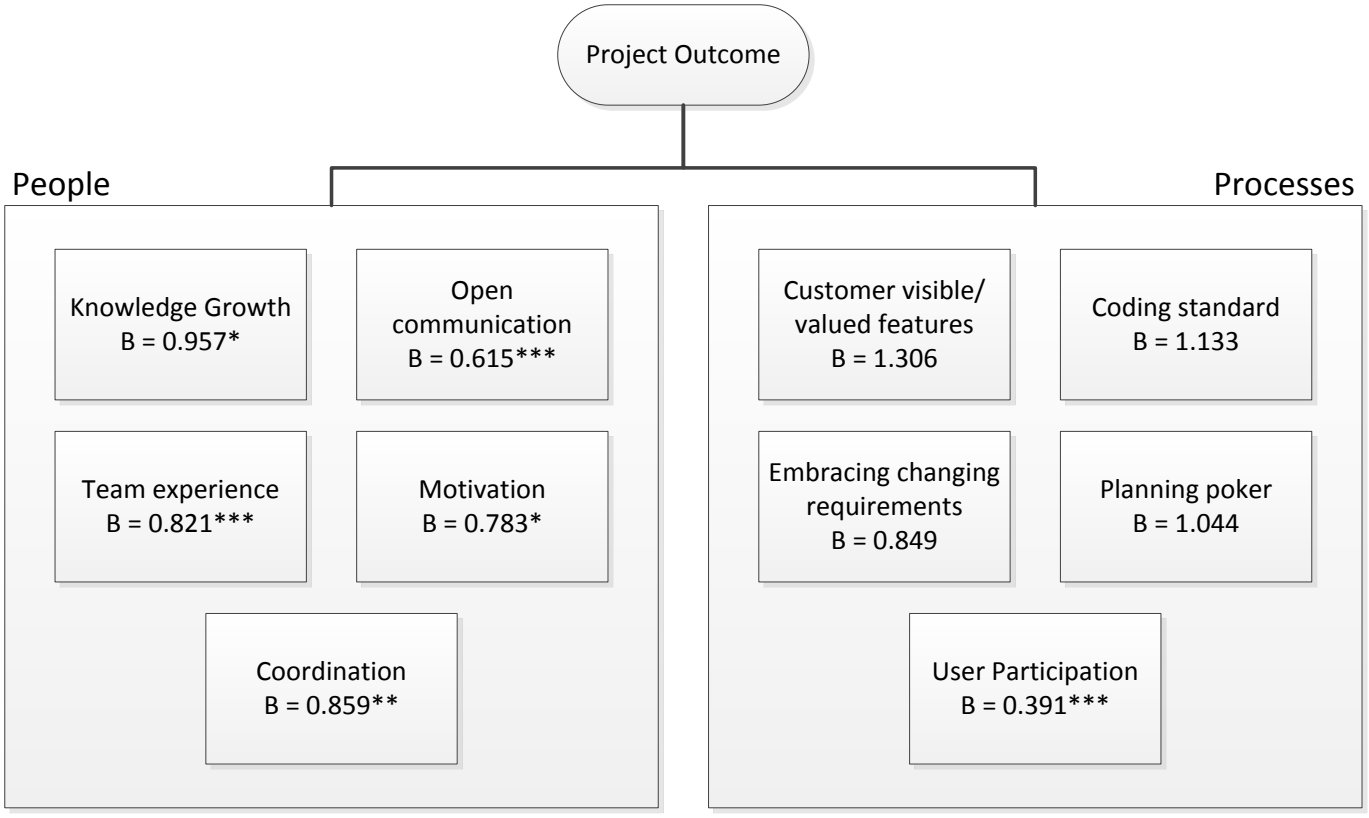


Figure 20. Significant associations of factors to project outcome

4.3 INTERVIEWS

4.3.1.1 Reporting process

Progress in all organisations is communicated via the person responsible for the project. For organisation 1 and 3 these are the scrum master / project manager. Organisation 2 has a project leader that reports to program management. In organisation 4 the product owner has meetings with the program manager to discuss the progress of the project. Burn-down charts are used by teams itself and not necessarily communicated to program or portfolio management. In organisation 1 the burn down charts are presented to program management.

Organisations 1, 2 and 3 use a traffic light system to show the status of the project. These are indicators from the perspective of a project manager. Formal documentation is small.

"The formal documentation reports are found in the systems and next to that there is a report of the project manager. He delivers the reports using documents." Program Manager, Organisation 3

"It is a rough estimation, there are no scientific calculations or whatever. The report I receive from them (Product Owners) *is purely: we are half way, on three quarters."* Program Manager, Organisation 4

Furthermore, information is mostly communicated orally through meetings.

"That (highlight report) I always discuss with the project manager in our bila. […] Meetings which I have at least 1 time in 2 weeks." Program Manager, Organisation 1

"Actually there is sufficient detail that we indeed can do oral." Program Manager, Organisation 3

"So I discuss that (status update) *amongst the Product Owners. They report amongst each other: we are this far…"* Program Manager, Organisation 4

Risks and issues are addressed not only during these meetings, but depending on the level of severity are communicated when they occur.

All organisations mention that progress is reported to program or portfolio management.

"…there I look at the highlight reports. I know the status. I know what is changed." Program manager, Organisation 1

"I receive progress reports monthly. In it is information about the usage of budgets, the usage in hours of the resources and further actually budgets in time and money." Portfolio manager, Organisation 2

"What were we going to do last period? What were we going to deliver? What are we going to deliver the next period? What did we make? What didn't we make? And risk and issues that kind of subjects are mentioned in there." Program manager, Organisation 3

"The report I receive from them (Product Owners) *is purely: we are half way, on three quarters."* Program Manager, Organisation 4

Organisation 3 mentions that the status of the project is done via meetings, due to the transparency of the report system. This could delay the process.

"If they are red then the whole world would see it. You do not want that, because then attention would be drawn. Then on at certain point you will get deliver insurance, a kind of quality audit, […] that just cost too much time." Program Manager, Organisation 3

A burn down chart is not commonly used to be communicated to program or portfolio management.

"That is only used in a team. Eventually, I will get the report of what they did in the last period and what they actually planned to do. So that comes from there (burn down chart)*."* Program manager, Organisation 3

Organisation 4 takes into the sourcing of people to projects that are having trouble or projects that need priority. This can be a permanent switch.

"Now and then, you look at who could we miss from that (team) and can he/she go to another team? […] Yes, it can be temporary, but it can also be permanent." Program Manager, Organisation 4

4.3.1.2 Challenges

The challenges in the reporting system consist of missing information that is desirable to be reported. Each organisation has other items that they miss, but find important to be reported. Organisation 1 and 2 are mentioning miss the quality assurance part of a report.

"The quality assurance picture I miss here and there and I don't know if the burn-up is going to be sufficient in its current state." Program Manager, Organisation 1

"What I strongly miss is an independent quality insurance function within the organisation I would almost say." Portfolio Manager, Organisation 2

Also, Organisation 1 finds that the work needed to be done is not visible enough yet.

"What the other work is what should be done is not visible enough, I think." Program Manager, Organisation 1

Organisation 3 finds it has troubles in communicating the progress within a sprint. It is desirable for the program managers such that he can prevent that projects result in never ending projects. Also the headcount is desirable to be reported.

"What troubles me [silence] then I think of when is a component really finished? You do know it already, but during a sprint it is difficult to estimate how far one really is. […] My biggest fear of that is that it results in never ending projects […], because it can still be improved." Program Manager, Organisation 3

"If I want to have more controle then I would look at what is your headcount is coming period." Program Manager, Organisation 3

Looking from a strategic perspective it is important to look at an important project that report confusing.

"If we see that an important project report confusingly then it's a signal for us, as is it going okay?" Portfolio Manager, Organisation 2

Organisation 4 has too few objective information, such that causes for delays or improvement of the process are difficult to assess.

"What is always the most difficult is that there is too few objective information. […] it is difficult to estimate if we don't run smooth here, this project runs out, are they just not doing well enough or did we estimate it wrong, is a team really up to speed or is the project just larger, is more coming from the business side or is complexity larger from IT side? […] And I know that there are systems like function points and others and that you can measure the velocity of a team, but you are not there yet with that." Program Manager, Organisation 4

Although object measures are needed, also the consensus is there to let the team focus on the implementation.

"…and we ask teams how many points they have done, what they have done in the sprint and that's it. We don't ask them difficult numbers or progress reports. No, that's it. So they need to spend their time on progress, not making reports." Program Manager, Organisation 4

5 DISCUSSION

5.1 ANALYSIS AND OPERATIONALIZATION OF THE MODEL

The aim of this study was to find how the model of McLeod and MacDonell (2011) could be operationalized in metrics for agile software development projects and programs/portfolios. The measurement of teamwork, within the People & Action dimension, is found to be associated with project outcome. However, the other two measurements in the dimensions Development Processes and Project Content were found to be inconclusive.

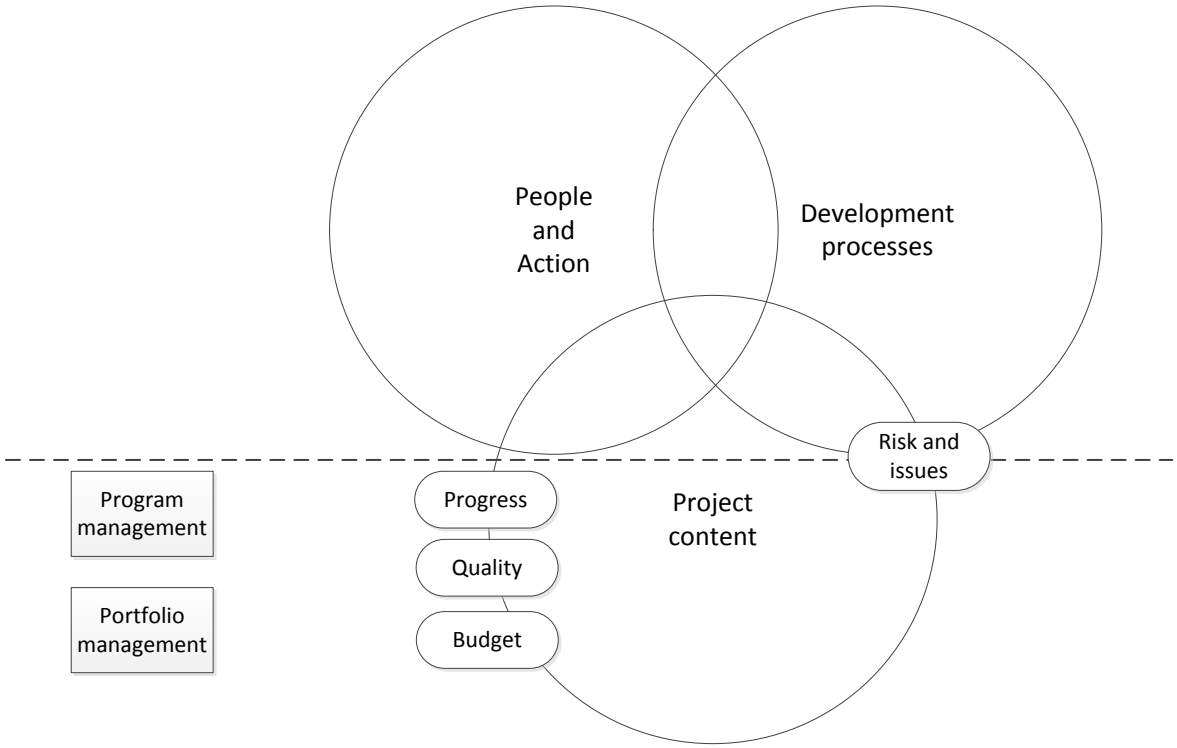


Figure 21. Reporting information for program and portfolio management

Figure 21 shows how the topics relate to the framework of McLeod and MacDonell (2011). These topics were mentioned by program and portfolio management in the interviews. The subject that was mentioned most during the interviews was progress. Other topics such as quality, budget, risks and issues were mentioned to be reported. Noticeably, there were no topics that could be placed in the dimension of People & Action and Development Processes.

For the suggested measurement tool it means that the most important information for program or portfolio management follows from the dimension Project Content. Risks and Issues is as explained by McLeod and MacDonell (2011) a part of project management which falls in the dimension Development Processes. Factors from People & Action are not mentioned in the interviews, but nonetheless important to take into account. An example of its importance is the usage of it for sourcing the right personnel to a project or to assess if the team is working well together. If the team is not effective, it could be a case to rotate personnel. The state of the current tool is incomplete for measuring the dimensions: Development Processes and Project Content.

People and Action Measuring teamwork the way it was done in this study can be reapplied in agile projects to measure the people dimension of the development team. The themes found to be influencing project outcome from the people and action dimension were knowledge growth, coordination, open communication and motivation. The categories social support and adaptation could not be proven to have an association with project outcome. Although the factors were found to be associated, more data is required to reduce the variance and to determine the actual strength of the associations.

In literature, user participation for agile projects was found to increase the satisfaction of the project (Subramanyam et al., 2010). There was a significant association between user participation and project outcome. Although the fact is that user participation increases success, when it is scored above 60% user participation was found to decrease success (Subramanyam et al., 2010). This is not accounted for in this study.

Senior management was found to have an association with project outcome. However, the effect of involvement, commitment and/or support of senior management is lower than for example the effects of the development team on project outcome. McLeod and

MacDonell (2011) mentioned that a low involvement of senior management might harm the project. Therefore, senior management should not be disregarded as a factor, but should receive less focus than other factors, such as the teamwork factors.

Development Processes The measurement based on agile practices for the process dimension is found to be inconclusive. Some practices are found to contribute to project outcome, such as: visible/valued customer features, embracing changing requirements and planning poker. However, other practices such as automated unit testing were not associated with project outcome.

From the results automated unit testing is found to have a statistically significant association with on-time delivery, on budget delivery and customer satisfaction. This relation was found to be negative. However, when comparing the answers with reality and literature, these associations found are not likely. In practise, a team partially integrated automated unit testing in their development process. Therefore, answers between the teams were different. Other teams that were not doing automated unit testing answered that they did apply it. So, these results give an inaccurate reflection of the reality.

Project Content Program or portfolio management is again mostly interested in progress. Quality and budget are also mentioned. Progress in the scrum method is commonly shown in a burn-up and/or burn-down chart. However, from the interviews most program managers found it sufficient to have an indication of progress. Even rough estimates were found to be acceptable. Organisation 3 indicated it missed the exact progress of the project, which the charts can help with. The traffic light system, in most cases, was used to signal the state of the budget. In all cases quality not measured for program or portfolio management. However, it was mentioned by program managers in 2 cases.

5.2 Challenges in portfolio/program management for project monitoring

Currently, program or portfolio managers received information based on informal meetings. This consisted of information of risks and issues, which was more detailed than when documented in the reports. Next to these meetings, reports were also used to pass information. Program managers indicated that this information was already made available during the informal meetings. The most important information program

management was interested in, was the progress of the project. Although, in some cases progress seemed not to be reported clear enough. Next to this, risks were also required to be communicated by the project managers.

The portfolio manager of organisation 2 found that an important project reported confusingly. This was a signal that action should be taken. Challenge here is to report more clearly, such that it is not necessary to rely on the signals but on the content of the reports.

In literature, in addition to the metrics mentioned earlier, it was found that portfolio management requires information for resource management and or strategic fit of a project (Krebs, 2008). These claims were not supported by this study, but also not contradicted, because it did not come to attention during the interviews. Resource management did came to attention in one interview where projects. The aspect quality was not mentioned in all interviews. The interviewees that did mention quality liked to receive more information for quality assurance.

5.3 Strengths and Limitations

One of the main strengths of this study is that this is the first study that incorporates a complete measurement system including all dimensions influencing a software development project. This study is a first step to continue to find the right method to such a measurement system. Measuring only within one dimension could miss opportunities to improve within the other dimensions. Also, several large companies operating in different areas were included, which resulted in a representative sample of the current state of agile practices.

This study has several limitations as well. First of all, due to a limitation in time and resources, the study is limited to reviewing a small number of projects and a limited amount of organisations. These limitations occurred due to the fact that this study was conducted as part of a master thesis. Besides, both questionnaires used have been timed to taking approximately 20 minutes per questionnaire. The length was determined by a pilot, where a few subjects filled in the questionnaire. The language used in the questionnaires was proven to contribute to the duration. However, the main cause of the

duration of the questionnaires was the number of influential factors measured in the questionnaires. This was also a reason for people to not complete the questionnaire or to refuse to participate in the study. Another reason to refuse was refusal due to not be able to book the time on a specific project code.

Furthermore, some properties within the model were not being reviewed, due to the lack of expert knowledge on how certain aspects of the models could be touched. The properties of the domains that are not described in this section were excluded from the study. Adding more questions concerning the dimensions of the model to the questionnaires will in general provide more detailed coverage, but cannot be covered due to the limitation of time. Although the *Institutional Context* dimension of the model is important, this dimension was not investigated in the current study. The reason for this is that it would take too much time finding measures to quantify the properties of this dimension.

All the projects that have been measured used Scrum as an agile method. Therefore, some projects have been measured at the end of a sprint while others have been measured at the end of a release. The reason to include a sprint as a deadline was because after each sprint a potentially shippable product should be available (Li, Moe, & Dybå, 2010). Due to the time limitation, this study has been set up in this manner. However, in order to get more reliable results, the measurement should be consistent in either sprints or releases. Here, the latter has the preference, due to strictness in achieving the deadline.

No conclusion could be drawn for the factors in the dimension Project Content due to confidential information. Information regarding hours budgeted, used hours, estimated costs and real costs could not be provided. Therefore, insufficient information was available to find associations and draw conclusions.

Some practices did not have an association with project outcome. A reason for this could be that participants did not know the specific practices although a short description was provided. Another reason could be that the opinion of some participants was that they were applying the practice to a certain extent. A result was that some practices were answered to be applied, but in reality they were not, leading to an overestimation of the

number of agile practices they applied. Furthermore, the rating of the project was self-reported, making it a subjective measure based on multiple factors such as experience. Practices could be scored higher, due to the fact that these kinds of questions provoke socially desirable answers. An attempt was made to reduce the influence of socially desirable answers by guaranteeing confidentiality at the top of the questionnaires. On the other hand, currently no objective measure that measures project outcome exists. In order to have an objective measure, SPICE practices were introduced to measure the maturity of the processes. However, respondents within projects gave varied answers of which processes were applied and which not.

Figure 20 shows the factors found to be associated with project outcome, based on the regression coefficient. However, the confidence intervals showed a large range. This means that the strength of the association is uncertain. The large range of the confidence interval is a result of the small number of participants.

6 RECOMMENDATIONS

The factors reported during the interviews all came from the Project Content dimension. These were progress, quality and budget. The present results suggest that teamwork measurement can be used as a tool to measure some of the aspects within the dimension People & Action. These results may have different levels of success for different organisations, because of other factors. Measuring the People & Action dimension based on the model of McLeod and MacDonell (2011) can help improve the execution of a project. Progress in certain cases is found to suffice when having a rough estimate. In other cases the use of a sprint burn-down chart and release burn-up chart should give more detailed information regarding progress. Budget in most cases is communicated through documents. When using burn-down or burn-up charts, the budget can even be shown in the chart. Quality can be determined by multiple different measurements, where suggestions are made by Krebs (2008).

Future research should investigate the agile practices further and it is also advisable to conduct research with a longitudinal study design to be able to determine causality. This study also requires an extension to a larger scale such that the number of participants for the questionnaires and interviews are greatly increased. Moreover, more variance between organisations is needed. Next to the number of respondents also the data concerning project details is needed in future measurements. Currently, project outcome is rated by the development team only. This can be extended by taking the average of the reported project outcome from different stakeholders in order to find a more reliable rating. Also, commercial success and customer satisfaction are rated by the development team. The customer can be involved to get a reliable rating. The number of questions included in the questionnaire should be limited, such that size is limited. Furthermore, from the model Institutional Context was not included in the study. However, it can influence the results and should also be investigated.

7 CONCLUSION

Using the model of McLeod and MacDonell (2011) in this study resolved a set of questions that can be used to measure projects. These questions were mainly found in the dimension People & Action. Little associations were found for using agile practices as a measure. Not enough data was retrieved to make statements for measuring Project Content.

Associations with project outcome were found with: coordination, open communication and knowledge growth. Found associations between agile practices and project outcome were: coding standard, customer visible/valued features, embracing changing requirements and planning poker. These associations are important to measure in order to find a connection with project outcome and therefore can function as a measurement tool. A higher project outcome is multidimensional and is greater than project success.

Program and Portfolio management mainly have factors within the dimension Project Content reported. However, it could be profitable when gathering information from all three dimensions for cases such as project portfolio management.

8 BIBLIOGRAPHY

(CMMI Product Team). (2006). CMMI ® for Development , Version 1 . 2, (August).

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods.

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. *25th International Conference on Software Engineering, 2003. Proceedings.*, 244–254. doi:10.1109/ICSE.2003.1201204

Amaratunga, D., Sarshar, M., & Baldry, D. (2002). Process improvement in facilities management: the SPICE approach. *Business Process Management Journal*, *8*(4), 318–337. doi:10.1108/14637150210434982

Ambler, S. W. (2005). A Manager's Introduction to The Rational Unified Process (RUP).

Ambler, S. W. (2006). Supersize Me. Retrieved April 17, 2013, from http://www.drdobbs.com/supersize-me/184415491

Anderson, D. J., & Schragenheim, E. (2003). *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results* (p. 336). Prentice Hall PTR.

Arent, J., & Nørbjerg, J. (2000). Software Process Improvement as Organizational Knowledge Creation : A Multiple Case Analysis, *00*(c), 1–11.

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, *17*(6), 337–342. doi:10.1016/S0263-7863(98)00069-6

Barton, A. B., Schwaber, K., Rawsthorne, D., Beauregard, C. F., Mcmichael, B., Mcauliffe, J., & Szalvay, V. (2005). Reporting Scrum Project Progress to Executive Management through Metrics, (January), 1–9.

Beck, K. (1999). Change with Extreme Programming, (c), 70–77.

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

Blomquist, T., & Müller, R. (2006). *Middle Managers in Program and Project Portfolio Management : Practices , Roles and Responsibilities*.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer*, *35*(1), 64–69. doi:10.1109/2.976920

Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, *22*(5), 30–39. doi:10.1109/MS.2005.129

Cabri, a., & Griffiths, M. (2006). Earned Value and Agile Reporting. *Agile 2006 (Agile'06)*, 17–22. doi:10.1109/AGILE.2006.21

Calkins, K. G. (2005). Correlation Coefficients. Retrieved July 19, 2013, from http://www.andrews.edu/~calkins/math/edrm611/edrm05.htm

Cerpa, N., & Verner, J. M. (2009). Why did your project fail. *Communications of the ACM*, *52*(12), 130–134.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*(6), 961–971. doi:10.1016/j.jss.2007.08.020

Cockburn, A., Bingham, K., Paulk, M. C., Mcmahon, P. E., Col, L., Palmer, G. A., Bowers, P., et al. (2002). Agile Software Development. *The Journal of Defense Software Engineering*, *15*(10).

Cockburn, A., & Highsmith, J. (2001). Agile Software Development: The People Factor, (November), 131–133.

Cohen, L., Manion, L., & Morrison, K. (2007). *Research Methods in Education* (6th ed., p. 634). 2 Park Square, Milton Park, Abingdon, Oxon: Routledge.

Cohn, M. (2004). *User Stories Applied*.

Cohn, M. (2005). *Agile Estimating and Planning*.

Conboy, K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research*, *20*(3), 329–354. doi:10.1287/isre.1090.0236

Conboy, K., & Fitzgerald, B. (2004). Toward a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines.

Cooper, R. G., Edgett, S. J., & Kleinschmidt, E. J. (1999). New Product Portfolio Management: Practices and Performance. *J PROD INNOV MANAG*, *6782*(99), 333–351.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. doi:10.1016/j.jss.2012.02.033

Downey, S., & Sutherland, J. (2012). Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9-10), 833–859. doi:10.1016/j.infsof.2008.01.006

Felici, M. (2004). Observational Models of Requirements Evolution.

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto, (August).

Glazer, H., Anderson, D., Anderson, D. J., Konrad, M., & Shrum, S. (2008). CMMI ® or Agile : Why Not Embrace Both !, (November).

Goldenson, D. R., & Emam, K. El. (1995). SPICE: An Empiricist's Perspective, *03*.

Gollapudi, K. V. V. G. B. (2004). Function Points or Lines of Code? An Insight.

Grenning, B. J. (2002). Planning Poker or How to avoid analysis paralysis while release planning, (April), 1–3.

Hartmann, D., & Dymond, R. (2006). Using Metrics and Diagnostics to Deliver Business Value, 2–7.

Hundermark, P. (2009). Do Better Scrum.

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and agile methods. *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, 520–525. doi:10.1109/CMPSAC.2004.1342889

Iqbal, A., & Abbas, S. S. (2011). Communication Risks and Best practices in Global Software Development.

Jakobsen, C. R., & Johnson, K. A. (2008). Mature Agile with a Twist of CMMI. *Agile 2008 Conference*, 212–217. doi:10.1109/Agile.2008.10

Jones, C. (2010). *Software Engineering Best Practices*.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A Framework fro Identifying Software Project Risks. *Communications of the ACM*, *41*(11), 76–83.

Kerth, N. (2000). The ritual of retrospectives, (October).

Kerzner, H. (2011). *Project Management Metrics, KPIs, and Dashboards*.

Kettunen, P. (2007). Agility with New Product Development Enterprise Agility, (June), 541–548. doi:10.1002/spip

Konrad, M. D., Paulk, M. C., & Graydon, A. W. (1995). An Overview of SPICE's Model for Process Management, (October), 291–301.

Krebs. (2008). *Agile Portfolio Management*.

Krebs, W., Kroll, P., & Richard, E. (2008). Un-Assessments Reflections by the Team, for the Team. *Agile 2008 Conference*, 384–389. doi:10.1109/Agile.2008.36

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History, (June), 47–56.

Li, J., Moe, N. B., & Dybå, T. (2010). Transition from a Plan-Driven Process to Scrum – A Longitudinal Case Study on Software Quality.

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, *85*(9), 2086–2095. doi:10.1016/j.jss.2012.04.005

McCabe, T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering*, *2*(4), 308–320.

McLeod, L., & MacDonell, S. G. (2011). Factors that affect software systems development project outcomes. *ACM Computing Surveys*, *43*(4), 1–56. doi:10.1145/1978802.1978803

Michael E Porter. (2008). *Competitive advantage: Creating and sustaining superior performance*.

Miller, A., & Carter, E. (2007). Agility and the Inconceivably Large. *Agile 2007 (Agile 2007)*, 304–308. doi:10.1109/AGILE.2007.14

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, *82*(11), 1869–1890. doi:10.1016/j.jss.2009.05.052

Misra, S., & Omorodion, M. (2011). Survey on agile metrics and their inter-relationship with other traditional development metrics. *ACM SIGSOFT Software Engineering Notes*, *36*(6), 1. doi:10.1145/2047414.2047430

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology*, *52*(5), 480–491. doi:10.1016/j.infsof.2009.11.004

Munassar, N. M. A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*, *7*(5), 94–101.

Nelson, R. R. (1987). Training End Users: An Exploratory Study. *MIS Quarterly*, (December), 547–560.

Nelson, R. R. (2005). Project Retrospectives: Evaluating Project Success, Failure and Everything in Between. *MIS Quarterly Executive*, *4*(3), 361–372.

NESMA. (2011). FPA according to NESMA and IFPUG; the present situation, 1–3.

Paulk, M. C. (2002). Agile Methodologies and Process Discipline.

Paulk, M. C., Curtis, B., Weber, C. V, & Chrissis, M. B. (1993). CMM for Software , Version 1.1, (February).

Paulk, M. C., Konrad, M. D., & Garcia, S. M. (1995). CMM Versus SPICE Architectures, (3), 7–11.

Salo, O. (2006). Enabling Software Process Improvement in Agile Software Development Teams and Organisations.

Schwaber, K. (1994). SCRUM Development Process, (April 1987), 10–19.

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum* (p. 158).

So, C. (2010). *Making Software Teams Effective: How practices Lead to Project Success Through Teamwork Mechanisms* (p. 195). Peter Lang GmbH.

So, C., & Scholl, W. (2009). Perceptive Agile Measurement: New Instruments for Quantitative Studies in the Pursuit of the Social-Psychological Effect of Agile Practices. *LNBIP 31*, 83–93.

SPICE. (1995). Consolidated product Software Process Assessment – Part 2 : A model for process management.

Stettina, C. J., & Heijstek, W. (2011). Five Agile Factors: Helping Self-management to Self-reflect. *EuroSPI*, 84–96.

Subramanyam, R., Weisstein, F. L., & Krishnan, M. S. (2010). User participation in software development projects. *Communications of the ACM*, *53*(3), 137. doi:10.1145/1666420.1666455

Sulaiman, T., Barton, B., & Blackburn, T. (2006). AgileEVM - Earned Value Management in Scrum Projects. *Agile Conference 2006 (Agile'06)*, 7–16. doi:10.1109/AGILE.2006.15

Sutherland, J. (2008). Agile Contracts: Money for Nothing and Your Change for Free. Retrieved February 20, 2013, from http://scrum.jeffsutherland.com/2008/10/agile-contracts-money-for-nothing-and.html

Sutherland, J., & Schwaber, K. (2011). The Scrum Guide, (October).

Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). Distributed Scrum: Agile Project Management with Outsourced Development Teams, 1–10.

The Standish Group. (2009). CHAOS Summary 2009: The 10 Laws of CHAOS, 1–4.

Tudor, D., & Walter, G. a. (2006). Using an Agile Approach in a Large, Traditional Organization. *Agile 2006 (Agile'06)*, 367–373. doi:10.1109/AGILE.2006.60

Verner, J. M., Cox, K., Bleistein, S., & Cerpa, N. (2007). Requirements engineering and software project success: an industrial survey in Australia and the US. *Australasian Journal of Information Systems*, *13*(1).

Verner, J. M., & Evanco, W. M. (2005). In-house software development: what project management practices lead to success? *Software, IEEE*, *22*(1), 86–93.

VersionOne. (2011). State_of_Agile_Development_Survey_Results.pdf.

Vinekar, V., Slinkman, C. W., & Nerur, S. (2006). Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management*, 31–43.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, *53*(1), 58–70. doi:10.1016/j.infsof.2010.08.004

Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287–1299. doi:10.1016/j.jss.2012.01.061

Ward-Dutton, N. (2011). Software Econometrics : Challenging assumptions about software delivery Podcast companion report.

West, D., Gilpin, M., Grant, T., & Anderson, A. (2011). Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today.

Williams, L. (2012). What agile teams think of agile Principles. *Communications of the ACM*, *55*(4), 71–76.

Yin, R. (2003). *Case Study Research* (3rd ed., p. 181). SAGE Publications.

Yip, J. (2011). It's Not Just Standing Up: Patterns for Daily Standup Meetings. Retrieved March 19, 2013, from http://martinfowler.com/articles/itsNotJustStandingUp.html

9 WORKS CITED

Oxford Dictionaries. (2012). (Oxford University Press) Retrieved 12 05, 2012, from http://oxforddictionaries.com/definition/english/metric

Alshayeb, M., & Li, W. (2004). An empirical study of system design instability metric and design evolution in an agile software process. *Science Direct*, Elsevier Inc.

Ambler, S. (2006). Supersize me. *Software Development, 14*(3), 46-48.

Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: steering from the edges. *Communications of the ACM, 48*(12), 85-89.

Basili, Caldiera, & Rombach. (n.d.). The Goal Question Metric Approach.

Beck, K. (2002). *Test Driven Development: By Example.* Addison-Wesley Professional.

Boehm, B., & Turner, R. (2005). Management Challenges to implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 30-39.

Boehm, B., Clark, B., Horowitz, E., & Westland, C. (1995). Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, 57 - 94.

Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization. *Computer, 36*(6), 74-78.

Goodman, P. (2004). *Software Metrics: Best Practices for Successful IT Management.* Rothstein Associates.

Highsmith, J. (2002). *Agile Software Development Ecosystems.* Addison-Wesley: Boston, Massachusetts.

Karlstorm, D., & Runeson, P. (2005). Combining agile methods with Star- Gate project management. *IEEE Software, 22*(3), 43-49.

Koch, A. (2005). *Agile Software Development: Evaluating the Methods for Your Organizations.* Northwood, Massachusetts: Artech House.

Larman, & Basili. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer Society*.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., & Kiefer, D. (2004). Agiles software development in large organizations. *Computer, 37*(12), 26-34.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Cahllenges opf migrating to agile methodologies. *Communications of the ACM, 46*(5), 72-78.

Palmer, S. R., & Felsing, M. (2001). *A practical guide to feature-driven development.* Pearson Education.

Reel, J. (1999). Critical success factors in software projects. *IEEE Software, 16*(3), 18-23.

Reifer, D., Maurer, F., & Erdogmus, H. (2003). Scaling agile methods. *IEEE Software, 20*(4), 12-14.

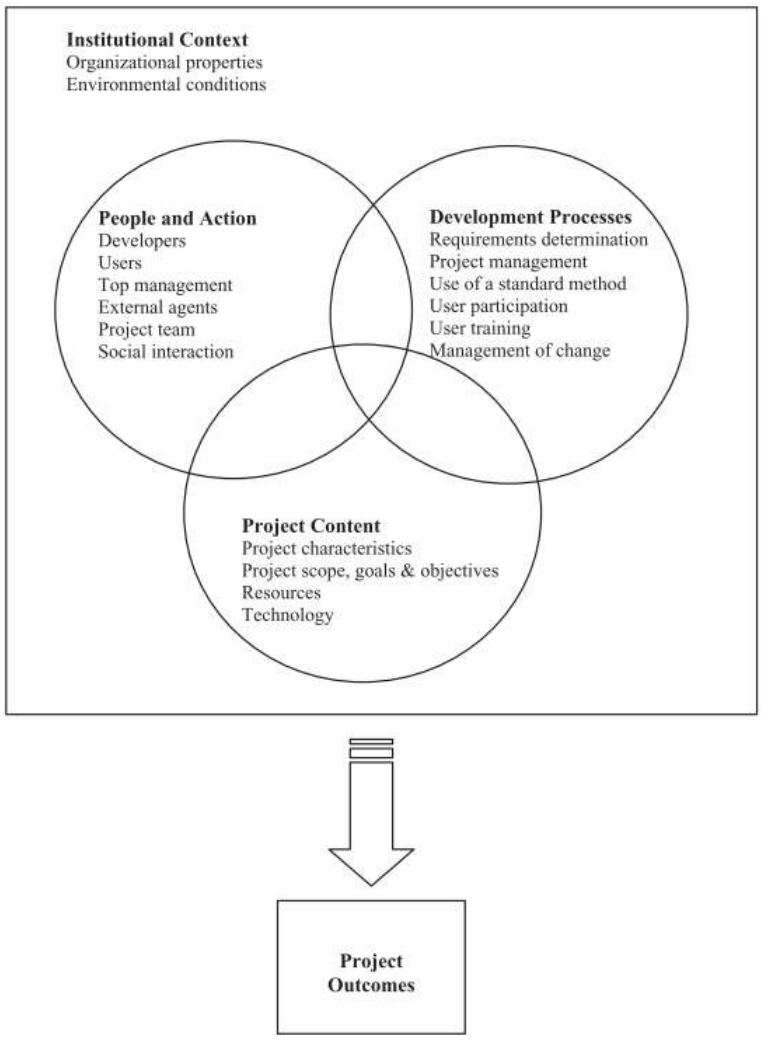
Schatz, B., & Abdelschafi, L. (2005). Primavera gets agile: A successful transition to agile development. *IEEE Software, 22*(3), 36-42.

Scotto, M., Sillitti, A., Succi, G., & Vernazza, T. (2006). A non-invasive approach to product metrics collection. *Journal of Systems Architecture*, 668-675.

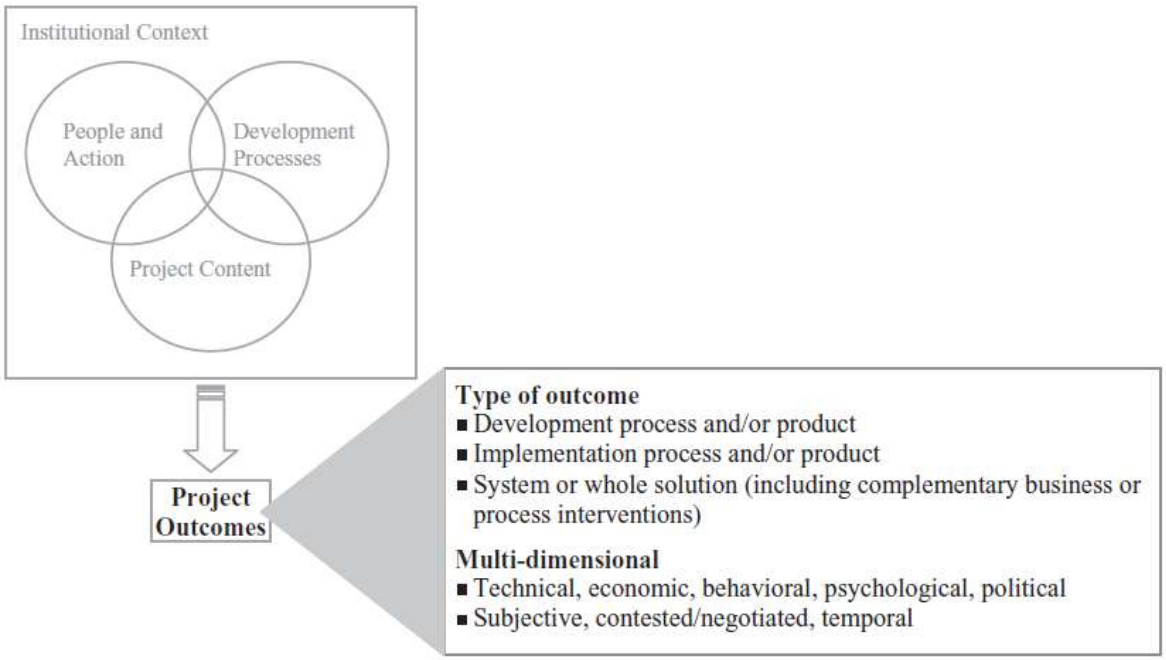
Vishwanath, A. (2012, 12 10). *Should we stop using Story Points and Velocity?* Retrieved from InfoQ: http://www.infoq.com/news/2012/12/stop_points_velocity

Williams, L., Brown, G., Meltzer, A., & Nagappan, N. (n.d.). Scrum + Engineering Practices: Experiences of Three Microsoft Teams.

APPENDIX A: FRAMEWORK OF MCLEOD AND MACDONELL (2011) IN DETAIL

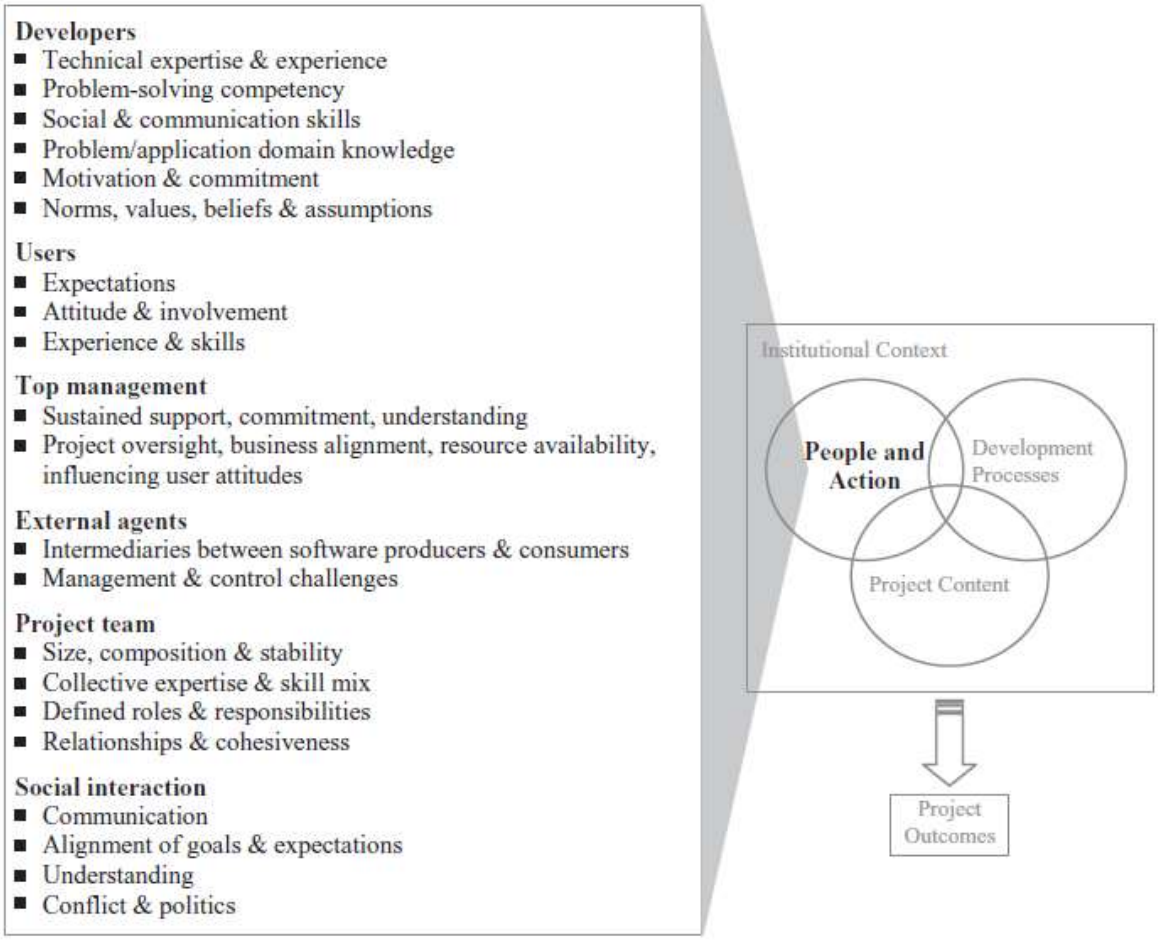


PROJECT OUTCOME



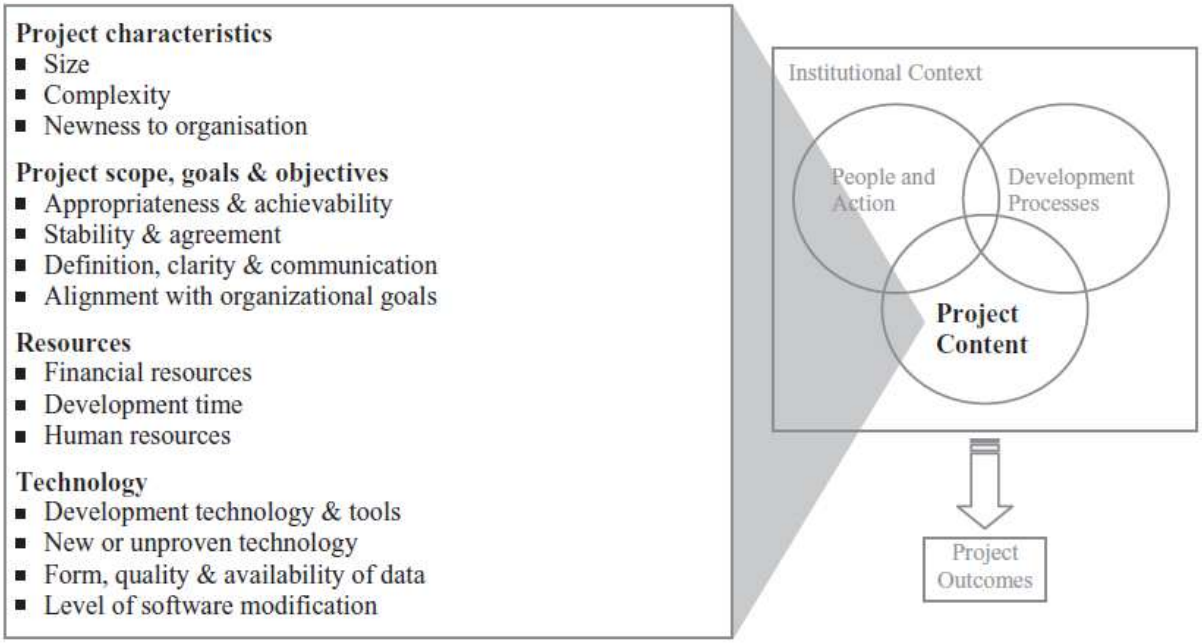
Project outcomes usually are described by means of project failure or project success. The outcomes are not two-dimensional but multi-dimensional. It is not only described by fail or success but by a combination of factors that are found in the figure above.

PEOPLE AND ACTION



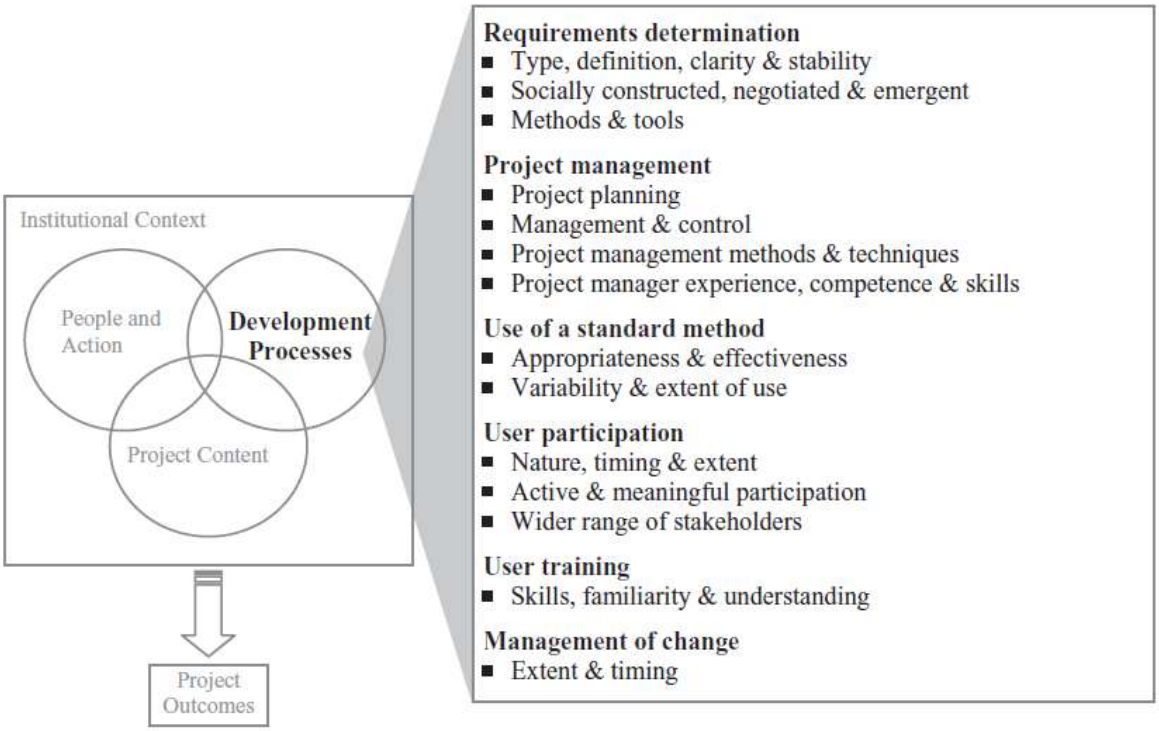
This domain describes the important factors that are influenced by the human nature. The stakeholders within this domain again influence the project in its own way or function. Influencing the project can be conscious or subconsciously and affect the project with a certain impact. This impact however is not described.

PROJECT CONTENT

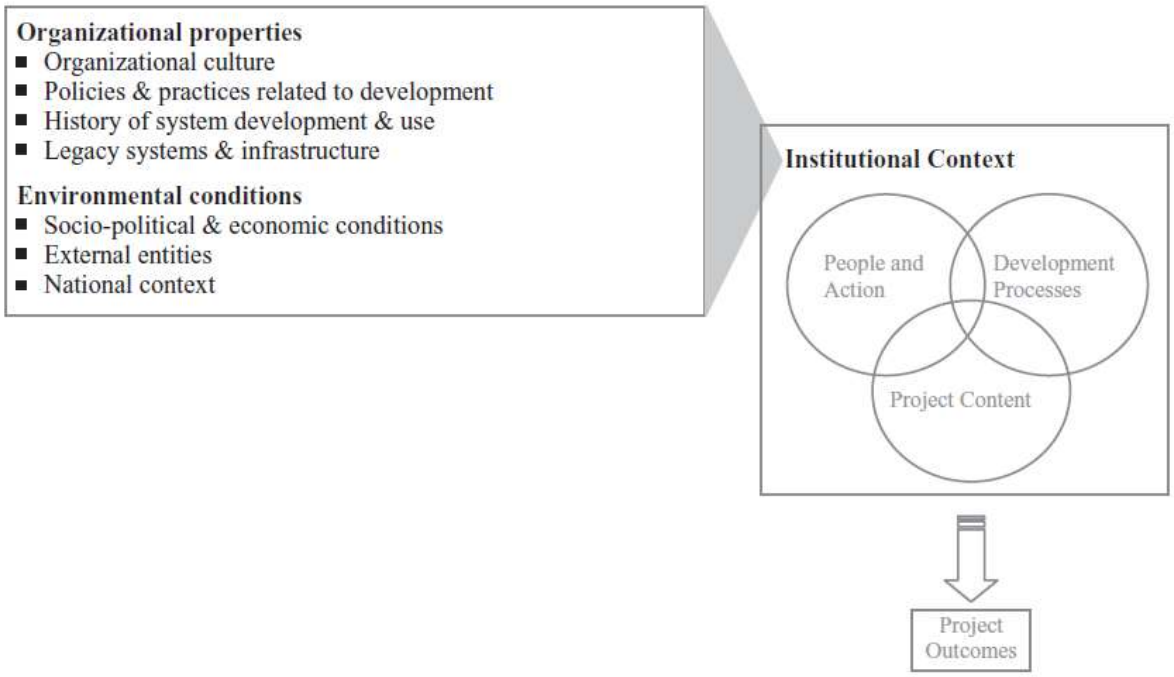


The Project Content is consists of several properties that are in general described by project characteristics, project scope goals and objectives, resources and technology. At the start of the project usually the “what” should be known. This is defined by the scope, goals and objectives of the project. The characteristics of a project can be described by how large, how complex and how new the project is to the organisation. Also projects can vary in the number of resources that is at disposal. the used technology that

DEVELOPMENT PROCESSES



INSTITUTIONAL CONTEXT



Although this dimension is not prominently included in the study it is an important factor that influences the behaviour and decisions that are made in the organisation, which then again influences the outcome of a project. The reason not to include this dimension in the current study is that it would take too much time finding measures to quantify the properties of the dimension. Furthermore, the lack of capability would also play a role here.

APPENDIX B: 12 PRINCIPLES OF THE AGILE MANIFESTO

ORIGINAL PRINCIPLES

The Agile Manifesto also mentions 12 principles where the development of software is based on (Fowler & Highsmith, 2001):

1. Satisfy the customer through early and continuous delivery of valuable software
2. Sustainable development is promoted, facilitating indefinite development
3. Simplicity is essential
4. Welcome changing requirements, even late in development
5. Deliver working software frequently
6. Working software is the primary measure of progress
7. Continuous attention to technical excellence
8. Business people and developers must work together daily
9. Face-to-face communication is the best method of conveying information
10. The team regularly reflects on how to become more productive and efficient
11. The best work emerges from self-organising teams
12. Build projects around motivated individuals

SUGGESTED PRINCIPLES

The study involving 326 respondents concludes the following principles to be applied (Williams, 2012):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements at the start of each iteration, even late in development; agile processes harness change for the customer's competitive advantage.
3. The whole team, from businesspeople through testers, must communicate and collaboratively work together throughout the project.
4. Build projects around empowered, motivated individuals with a shared vision of success; give them the environment and support they need, clear their external obstacles, and trust them to get the job done.
5. The most efficient, effective method for conveying information to and within a development team is through synchronous communication; important decisions are documented so are not forgotten.

6. Valuable, high-quality software is the primary measure of progress at the end of each short timeboxed iteration.
7. Agile processes promote sustainable development. The whole team should be able to maintain a reasonable work pace that includes dedicated time for exploration, visioning, refactoring, and obtaining and responding to feedback.
8. Continuous attention to technical excellence and good design enhances agility.
9. Simplicity—the art of maximizing the amount of work not done—is essential.
10. The best architectures, requirements, and designs emerge from self-organizing teams guided by a vision for product release.
11. With each iteration, the team candidly reflects on the success of the project, feedback, and how to be more effective, then tunes and adjusts its plans and behavior accordingly.

APPENDIX C: SPICE BASELINE PRACTICE GUIDE & BEST SOFTWARE- AND AGILE PRACTICES

SPICE BASELINE PRACTICE GUIDE (BPG)

The following 5 tables are the baseline practices for different process categories within an organisation, practices specified for software engineering or management activities, which are identified by ISO 15504. In order to specify the maturity of a process these practices can be identified by the assessment of SPICE to be *not-performed, performed-informally, planned-and-tracked, quantitatively-controlled* or *continuously-improved* (Konrad et al., 1995). A more detailed list of the engineering practices are found questionnaire 2 in Appendix D: Questionnaires and interviews.

Customer-Supplier
Acquire Software Product and/or Service
Establish Contract
Identify Customer Needs
Perform Joint Audits and Reviews
Package, Deliver, and Install the Software
Support Operation of Software
Provide Customer Service
Assess Customer Satisfaction

Engineering
Develop System Requirements and Design
Develop Software Requirements
Develop Software Design
Implement Software Design
Integrate and Test Software
Integrate and Test System
Maintain System and Software

Project
Plan Project Life Cycle
Establish Project Plan
Build Project Teams
Manage Requirements
Manage Quality
Manage Risks
Manage Resources and Schedule
Manage Subcontractors

Support
Develop Documentation
Perform Configuration Management
Perform Quality Assurance
Perform Problem Resolution
Perform Peer Reviews

Organization
Engineer the Business
Define the Process
Improve the Process
Perform Training
Enable Reuse
Provide Software Engineering Environment
Provide Work Facilities

AGILE PRACTICES

The following table contains the practices that was researched to be most used in industry (Williams, 2012). The list is ordered by most practiced to least practised by the respondents of the study.

Agile Practices
Continuous integration
Short iterations (30 days or less)
Done criteria
Automated tests run with each build
Automated unit testing
Iteration reviews/demos
Potentially shippable features at the end of each iteration
Whole multidisciplinary team with one goal
Synchronous communication
Embracing changing requirements
Features in iteration are customer-visible/customer-valued

Prioritized product backlog
Retrospective
Collective ownership of code
Sustainable pace
Refactoring
Complete feature testing done during iteration
Negotiated scope
Stand up/Scrum meeting
Timeboxing
Test-driven development unit testing
Just-in-time requirements elaboration
Small teams (12 people or less)
Emergent design

Configuration management	10-minute build
Daily customer/product manager involvement	Task planning
Release planning	Coding standard
Test-driven development acceptance testing	Kanban
Team documentation focuses on decisions rather than planning	Acceptance tests written by product manager
Informal design; no big design up front	Pair programming
Co-located team	Burndown charts
Team velocity	Code inspections
Requirements written as informal stories	Design inspections
	Planning Poker
	Stabilization iterations

SOFTWARE ENGINEERING PRACTICES

In the following table the best software engineering practices are given. The set of best practices spans the entire life cycle from the day a project starts until the day that the application is withdrawn from service (Jones, 2010).

Best software engineering practices	
Minimizing harm from layoffs and downsizing	Outsourcing software applications
Motivation and morale of technical staff	Using contractors and management consultants
Motivation and morale of managers and executives	Selecting software methods, tools, and practices
Selection and hiring of software personnel	Certifying methods, tools, and practices
Appraisals and career planning for software personnel	Requirements of software applications
Early sizing and scope control of software applications	User involvement in software projects
	Executive management support of software applications
	Software architecture and design

Software project planning
Software project cost estimating
Software project risk analysis
Software project value analysis
Canceling or turning around troubled projects
Software project organization structures
Training managers of software projects
Training software technical personnel
Use of software specialists
Certification of software engineers, specialists, and managers
Communication during software projects
Software reusability
Certification of reusable materials
Programming or coding
Software project governance
Software project measurements and metrics
Software benchmarks and baselines
Software project milestone and cost tracking
Software change control before release

Configuration control
Software quality assurance
Inspections and static analysis
Testing and test library control
Software security analysis and control
Software performance analysis
International software standards
Protecting intellectual property in software
Protection against viruses, spyware, and hacking
Software deployment and customization
Training clients or users of software applications
Customer support after deployment of software applications
Software warranties and recalls
Software change management after release
Software maintenance and enhancement
Updates and releases of software applications
Terminating or withdrawing legacy applications

APPENDIX D: QUESTIONNAIRES AND INTERVIEWS

INTERVIEW 1 (I1)

Introduction

Introducing myself by telling me who I am, my background and education. Informing the participant about my research and how this interview contributes to the research. Ask for approval to record the interview. Emphasize that all data will be made anonymous.

General Information

- What is your name?
- What is your working/educational background?
- What is your current function?
- What are you currently working on?

Project reporting

- In what projects are you currently involved?
- How do the projects fit the portfolio / program?
- What is the scope / goal of the respective projects? How are the goals communicated to the development teams?
- Is it possible to have a (anonymous) copy of the project details?
- How do the teams report to you?
- Can you write down on a piece of paper the exact process steps? *(Provide pen and paper and ask participant to write down process as detailed as possible step-by-step)*
- Which information do you correctly receive within the reports?
- Which information is currently missing and would you like to receive?
- What is currently going well in the reporting process?
- What are the challenges in the reporting process?
- What actions do you follow when you have received the reports?
- How do you store the data of the reports?

Program – Portfolio management process

- Does your organisation have a central administration of projects? *(e.g. a project/program/portfolio management office)*
- What is your process of prioritising allocation (of resources), monitoring and reviewing projects?
    - What models do you use for balancing and prioritising projects within portfolio/program (e.g. financial models like ROI, NPV and IRR, strategic approaches?
    - Could you please write down a step-by-step description of the portfolio/program process as detailed as you remember? (How is the project portfolio filled and maintained)
- Can you draw on a piece of paper the portfolio process? *(Provide pen and paper and ask participant to write down process as detailed as possible step-by-step)*
- Are you reviewing milestone, cost, and risk status reports?
    - If yes, how much of your time do you spend on this?
    - If no, who is reviewing the reports?

- Are you assigning key executives to oversight, governance, and project director roles?
    - If no, who is assigning these roles?
- Are you determining if overruns or delays have reduced the ROI below corporate targets?
    - If yes, what are your actions based on the results? How much time do you spend on the process?
    - If no, who is responsible?

Software development process

- Which process (methodology e.g. Scrum) does your organisation follow for the software projects?
    - Could you please write down the development process? *(Provide pen and paper and ask participant to write down process as detailed as possible step-by-step)*
    - Do you have documentation (illustration or text) of your process that I could take with me?
- How did your organisation choose/deploy the current method (practices)?
- Which agile practices are applied (e.g. iteration, stand-up meetings, customer involvement, continuous integration)?
- Are experiences (processes) from prior and current project collected and applied? How?
    - Which experience collection methods do you apply?
- Are you satisfied with the process?
- What are the current challenges?

Management involvement in projects

- Are you involved in the portfolio or program meetings? Who else is involved? How often do these meetings take place? What will be discussed?
- Are you responsible for the budgeting of the software development projects?
- Are you responsible for the budget of the software development projects?
    - If yes, what is your role in this? Are there others roles involved?
    - If no, who is? Can you try to explain his/her role? Are there others roles involved?
- What are the project metrics that are collected? Where do you use them for?
- What artefacts are used (e.g. excel sheets, documents, etc)?
- How supportive are you in the projects? Can you give an example of your support?
- How do you distribute your attention over the projects?

INTERVIEW 2 (I2)

General Information

- What is your name?
- What is your educational background?
- What working experience do you have?
- What is your role in the organisation?

Project characteristics

- Can you describe the project in short?
- How did the project come to place?
- Is the size of the software measured? If yes, what is the size and in which unit is it measured? If no, what did the project cost in total?
- Is the complexity of the software measured? If yes, what is the complexity of the software? If no, what is the estimation of complexity of the project?
- Is the quality of the software measured? If yes, what is the result and in which unit is it expressed? If no, how was the software tested?

Project scope

- Does/Did the project have a well-defined scope? How is this measured?
- Did the scope increase during the project? What were the reasons?

Project resources

- How many hours are/were estimated for this project?
- How many hours are/were needed for this project?
- What is/was the estimation of the project costs?
- What is/was the amount budgeted for the project?
- What does/did the project cost?
- Is/Was the deadline of the delivery of the project met? If no, is the project finished or what were the reasons the deadline could not be met?
- How many people are/have work(ed) on the project?
- How many people are/were developing the software?
- How many development teams worked on the project?
- Could you write down the names of the teams working on the project? How many people were there on each team?

Project management process

- Which process does your organisation follow to develop software projects? (Provide pen and paper and ask participant to write down process as detailed as possible step-by-step)

Software metrics

- Which software metrics are you using and what is the purpose of measuring these?
- Does the current software metrics provide acceptable feedback for improvements?
- What software metrics do you currently miss?

QUESTIONNAIRE 1 (Q1)

Dear participant, this questionnaire will contribute to the research that tries to assess the knowledge concerning project outcome. It takes into account the aspects that influence the project at team level, for example: communication. All the information you provide will be kept confidential and anonymous.

Universiteit Leiden

*All questions reflect on the mentioned project. This questionnaire contains **5 pages of questions** printed **double sided** and takes approx. **20 minutes** to fill in. Please answer the questions honestly and to the best of your knowledge and complete all questions. Thank you!*

General Information

What was the project called/identified as? (e.g. Project Y, Project Z, etc)
.....

If there were multiple teams, in which team were you for this project? (e.g. team X, team Y, etc)
.....

What is your role in the team? (e.g. developer, product owner, etc)
.....

What was the duration of this sprint/phase (in weeks)? (e.g. 2 weeks, 3 weeks, etc)
.....

Teamwork

Goal Commitment								
Commitment on goals is the motivation of people to stick to the goals regardless of difficulties or obstacles encountered. On an agile project, the iteration goals are represented by the scope items to be implemented by iteration end. How did you perceive the following aspects of goal commitment in your project?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
It was very important to me that my team met the							

assigned iteration goals.							
I was strongly committed to pursuing my team's iteration goals							
The iterations goals appeared reasonable to me.							
I didn't care if my team achieved its iteration goals or not							
I was highly motivated to help my team meet our assigned iteration goals							
The iterations goals appeared realistic to me							

Social Support							
Support from people at work can consist of actual help in completing work assignments, but also in social interaction conveying empathy. How did you perceive social support from your colleagues on this project/release?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
Did you get on well with your colleagues?							
Were your colleagues friendly towards you?							
Was there a good atmosphere between you and your colleagues?							
On this project, could you count on your colleagues when you encountered difficulties in your work?							
Did you feel highly appreciated by your colleagues on this project?							

Open communication							
Openness of group communication represents the degree to which people can talk freely about critical issues, e.g. their own mistakes. On this project/release, how did you perceive the following aspects of communication among team members excluding the customer)?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
People had to watch what they say when bringing up sensitive issues.							
Team members hesitated to report mistakes.							

Team members hesitated to raise attention within the team to problems							
Important information was withheld							
Team members brought bad news to the team with reservation							
I could get the most work-relevant information from other team members only if I specifically asked for it.							

Adaptation							
Adaptation is the degree of mutual adjustment between the customer and the team: As the customer learns from the team's constraints on feasibility, the team grows a better understanding of the changing customer requirements, and tries to adapt to these changes in the following iteration. From the viewpoint of your team, how did you perceive the following aspects of adaptation in your project?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
Our team adapted the planned or implemented scope to changing customer requirements							
We could modify the implemented scope according to customer requests with ease.							
We were very flexible in the approach we used to handle changing customer requirements.							
We could effectively change our scope when changed customer priorities or requirements made it necessary.							

Coordination Capability							
Division of work and customer interaction are indispensable to agile software development. Unfortunately, this sometimes leads to problems in the execution of projects because of inefficient coordination with the customer or within the development team.	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
How often have internal decision making processes stagnated?							
How often have discussions circled endlessly?							
How often did executed activities lack coordination with the customer?							
To what extent did the implementation of decisions turn out to be mistaken?							
How often were strategic decisions implemented without adapting them to the situation or context?							

While implementing a decision, how often did people execute it deliberatly not according to the decision's intention?							

Knowledge Growth							
To what extent do the following statements describe your project/release in regards to knowledge growth of the technical team as a whole?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
We gained a profound understanding of where the real problems lay in this project							
We discovered mistakes early							
We improved processes to a high extent							
We integrated and leveled ideas and experience from others (internal and external).							
We knew when and how the customer changed requirements or priorities.							
We had a close understanding of any changing customer requirements.							

Team Performance							
How successful is/was the project/release according to your opinion?	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
Deliver on-time							
Deliver within budget							
Deliver high quality							
Adapting to changing customer requirements							
Customer satisfaction							
Commercial success							
Lessons learned							
Project success overall							

Additional questions

The scale for the following questions are: very poor 1 – 10 excellent

How would you rate your domain knowledge on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

……

……

How would you rate your motivation on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

……

……

How would you rate the experience of the team on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

……

……

You can leave any additional comments or thoughts here

You have reached the end of the questionnaire. Thank you for your participation!

QUESTIONNAIRE 2 (Q2)

Dear participant, this questionnaire will contribute to the research that tries to assess the knowledge concerning project outcome. It looks at the aspects that influence the project, by looking at which practices are used and the execution of the development process. All the information you provide will be kept ***confidential and anonymous****.*

All questions reflect on the mentioned project. This questionnaire contains ***8 pages of questions*** *printed* ***double sided*** *and takes approx.* ***20 minutes*** *to fill in. Please answer the questions honestly, to the best of your knowledge and complete all questions. Thank you!*

General information

What was the project called/identified as? (e.g. Project Y, Project Z, etc)
.....

What is your role in this project? (e.g. developer, product owner, etc)
.....

If the project has ended/had a release recently, when was this? (e.g. a few days ago, a week ago, etc)
.....

Applied practices

This section covers which practices are used in the project. The scale of this section is the following:

Yes; No, but should apply it; No, not necessary; I don't know

Requirement determination: methods & tools *Do you perform the following requirements practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
1. Joint client/vendor change control board (e.g. containing clients and/or vendors that make decisions on changes for software projects)				
2. Domain experts for changes to specific features				

3. Requirements traceability is present (i.e. ensuring that changes to a requirement can be traced back until the source)				
4. Multiple releases of requirements changes (i.e. requirements changes are released in batches)				
5. Utilization of automated requirements analysis tools				
6. Careful analysis of the features of packages that will be part of the application				
7. Joint application design (JAD) for initial requirements gathering (i.e. knowledge workers and IT specialists meet to define and review business requirements)				
8. Quality function deployment (QFD) for quality requirements (i.e. a method to transform user demands into design quality)				
9. Security analysis and vulnerability prevention				
10. Prototypes for key features of new applications (e.g. to show progress to a user)				
11. Full-time user involvement for Agile projects (i.e. committed users)				

Requirement determination: methods & tools *Do you perform the following requirements practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
12. Mining legacy applications for requirements and business rules for new projects				
13. Clear and understandable requirements				
14. Formal requirement inspections with both users and vendors				

Agile development *Do you perform the following agile practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
1. Continuous integration (i.e. integrate new features to current products every iteration)				
2. Short iterations (<30 days) (e.g. to ensure frequent stakeholder interaction)				
3. Done criteria / Definition of "Done" (i.e. a definition of done created to know when a function is really completed)				

4. Automated tests run with each build				
5. Automated unit testing				
6. Iteration reviews/demos (i.e. review session of features built in latest iteration)				
7. Whole multidisciplinary team with one goal (i.e. a team with individual wide knowledge having one goal)				
8. Features in iteration are customer-visible/valued (i.e. new features are made clear to the customer and are visible/valuable)				
9. Prioritized product backlog (i.e. the backlog is ordered from most important feature first to least important)				
10. Retrospective (i.e. to reflect on the last iteration for process improvement)				
11. Collective ownership of code (i.e. everyone in the team is allowed to access/alter the source code)				
12. Stand up/Scrum meeting (i.e. daily meeting to ensure communication in the team)				
13. Small teams (12 people or less) (e.g. to prevent freeriding or lack of overview)				
14. Co-located team (e.g. the whole team works in the same room)				
15. Coding standard (i.e. a standard for the source code for the team)				
16. Pair programming (i.e. two developers working on the same item with one keyboard)				
17. Burndown charts (i.e. a chart showing the progress of finished story points set against schedule)				
18. Requirements written as informal stories (e.g. as [role] I want [function] to [reason])				

Agile development	Yes	No, but	No, not	I don't

Do you perform the following agile practices in this project?		should apply it	necessary	know
19. Embracing changing requirements				
20. Test-driven development acceptance testing				
21. Code inspections (i.e. a peer reviews the written source code)				
22. Planning Poker (i.e. collaborative method for better estimates of user stories)				

Software engineering & programming *Do you perform the following software engineering and programming practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
1. Early sizing and scope control				
2. User involvement in software projects (e.g. to do review business rules, requirements, design, documents, prototypes, do defect reporting and acceptance testing)				
3. Selecting software methods, tools and practices (e.g. requirements inspections, design inspections, formal risk analysis)				
4. Having a software architecture and design				
5. Selection of reusable code from certified sources, before starting to code				
6. Planning and including security topics in code (e.g. including secure languages such as E)				
7. Version control (i.e. managing updates of code via by creating versions)				
8. Having clear and structured code				
9. Formal code inspections of all modules				
10. Clear and relevant comments in the source code				
11. Re-inspection of code after significant changes or updates				

User training *Do users get offered the following trainings for this project?*	Yes	No, but should apply it	No, not necessary	I don't know
1. Tutorial (e.g. individually taught by instructor with few instructional material)				
2. Courses/seminars (e.g. taught by instructor that determines course content and provides instructional material)				
3. Computer-aided instruction [CAI] (e.g. computer-based tutorial or drill and practice)				
4. Interactive training manual (e.g. a combination of tutorial and CAI)				

User training *Do users get offered the following trainings for this project?*	Yes	No, but should apply it	No, not necessary	I don't know
5. Resident expert (e.g. like a tutorial, but the training is initiated by the user)				
6. Help component (e.g. error messages with in some cases explanations)				
7. External (e.g. courses as in MBA-programs or vendor/independent seminars)				

SPICE[1] Practices *Do you perform the following SPICE practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
1. **Specifying system requirements** Determine the required functions and capabilities of the system and document in a system requirements specification.				
2. **Describing system architecture** Establish the top-level system architecture, identifying elements of hardware, software and manual operations.				
3. **Allocating the system requirements to the top-level system architecture, including software** Allocate all system requirements to the elements of the top level system architecture, including software.				
4. **Determining the release strategy** Prioritize the system requirements and map them to future releases of the system.				
5. **Determinining and documenting the software requirements** Determine the software requirements and document in a software requirements specification.				

[1] SPICE stands for "Software Process Improvement and Capability dEtermination"

6. **Analysing the software requirements for correctness** Analyse the software requirements for correctness.				
7. **Determine the impact of the software requirements on the operating environment**				
8. **Evaluating requirements with customer** Communicate the software requirements to the customer, and revise if necessary, based on what is learned through this communication.				
9. **Updating requirements for next iteration** After completing an iteration of requirements, design, code, and test, use the feedback obtained from use to modify the requirements for the next iteration.				
10. **Developing a software architectural design that describes the top-level structure and identifies its major components** Transform the software requirements into a software architecture that describes the top-level structure and identifies its major components.				
11. **Designing internal and external interfaces at top level** Develop and document a top-level design for the external and internal interfaces.				
12. **Developing a detailed design** Transform the top-level design into a detailed design for each software component.				
13. **Establish traceability between the software requirements and the software designs**				
14. **Developing software units including code, data structure and database**				
15. **Developing and documenting unit verification procedures** Develop and document procedures for verifying that each software unit satisfies its design requirements.				
16. **Verifying and documenting the software units based on the design requirements** Verify that each software unit satisfies its design requirements and document the results.				

SPICE Practices *Do you perform the following SPICE practices in this project?*	Yes	No, but should apply it	No, not necessary	I don't know
17. **Determining the regression test strategy** Determine the conditions for retesting aggregates (collections) against their tests should a change in a given software unit be made.				
18. **Building collections of software units** Identify aggregates of software units and a sequence or partial ordering for testing them.				

19. **Developing tests for the collections indicating input data and acceptance criteria** Describe the tests to be run against each software aggregate, indicating input data and acceptance criteria.				
20. **Testing the software collections against the test criteria and documenting the results** Test each software aggregate ensuring that it satisfies the test criteria, and document the results.				
21. **Developing tests for software indicating software requirements being checked, input data, and acceptance criteria**				
22. **Testing the integrated software against the software requirements and documenting the results**				
23. **Building collections of system elements** Identify aggregates of system elements and a sequence or partial ordering for testing them.				
24. **Developing tests for the collections indicating input data, system components needed to perform the test, and acceptance criteria**				
25. **Testing system collections and documenting the results** Test each system aggregate ensuring that it satisfies its requirements, and document the results.				
26. **Developing tests for system indicating system requirements being checked, input data, and acceptance criteria** Describe the tests to be run against the integrated system, indicating system requirements being checked, input data, and acceptance criteria.				
27. **Testing integrated system satisfying the system requirments and documenting the results**				
28. **Determine maintenance requirements** Determine the system and software maintenance requirements, identifying the system and software elements to be maintained, and their required enhancements.				
29. **Analyze user problems and enhancements** Analyse user problems and requests and required enhancements, evaluating the possible impact of different options for modifying the operational system and software, system interfaces, and requirements.				
30. **Determine modifications for next upgrade** Based on the user problems and requests and required enhancements analyses, determine which modifications should be applied in the next system or software upgrade, documenting which software units and other system elements and which documentation will need to be changed and which tests will need to be run.				
31. **Implement and test modifications** Use the other engineering processes, as appropriate, to implement and test the selected modifications, demonstrating that the unmodified system and software requirements will not be compromised by the upgrade.				

32. **Upgrade user system** Migrate the upgraded system and software with applied modifications to the user's environment, providing for, as appropriate: parallel operation of the previous and upgraded systems, additional user training, support options and retirement of the previous system.				

Process perception

This section covers how you perceived the development process. The scale of this section is the following:

Never > Very rarely > Rarely > Occasionally > Frequently > Very frequently > Always

Requirement determination construction *The requirement determination process shows how the functions constructed for the functionality in the project. How is does this process follow in this project?*	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
1. Do you identify and consult all likely sources of requirements, system stakeholders?							
2. Do you look for constraints in the domain?							
3. Do you collect requirements from multiple viewpoints?							
4. Do you use language simply, consistently and concisely for describing requirements?							
5. Do you record requirements traceability from original sources?							
6. Do you record requirements reasons in order to improve requirements understanding?							
User participation *In agile projects users are more likely to be involved in the development process. Feedback and questions are can then gathered and answered directly. How/ Where did you perceive the participation of the user in the project?*	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
1. To what extent did a user participate in project scoping and prioritization of requirements?							
2. To what extent did a user participate in responding and providing inputs to product prototypes?							
3. To what extent did a user participate in design meetings?							
4. Did users make adequate time available for the requirements gathering?							
5. In our projects, customers closely collaborate with the development team members							

Project management *One of the tasks of the project manager is to have the project be delivered on time and within budget. Several methods can be applied to ensure this.*	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
1. The project manager was involved in making initial cost and effort estimates.							
2. The delivery decision was made with appropriate requirements information.							
3. The effort and schedule estimates were good.							
4. The developers were involved in making estimates.							
5. The project had adequate staff to meet the schedule.							
6. Staff was added to meet an aggressive schedule.							

Senior management *Projects are affected by the involvement of senior management. How did senior management interact with this project?*	Never	Very rarely	Rarely	Occasionally	Frequently	Very frequently	Always
1. Senior management was committed to this project.							
2. Senior management was supportive in this project.							
3. Senior management was involved in this project.							

Project and result rating

This section covers how you would rate the project outcome and the outcome of the product. The scale of this section is the following:

very poor 1 – 10 excellent

How would you rate the project in general on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

……

How would you rate the result (product/software) delivered on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

……

Additional questions

This section contains an additional question and a possibility to write down extra thoughts concerning this questionnaire.

The scale of this question is the following:

very poor 1 – 10 excellent

How would you rate your experience on a scale of 1 – 10, where 10 is excellent?	1	2	3	4	5	6	7	8	9	10
[Tick one of the boxes to the right and please elaborate your choice below]										

Please elaborate

……

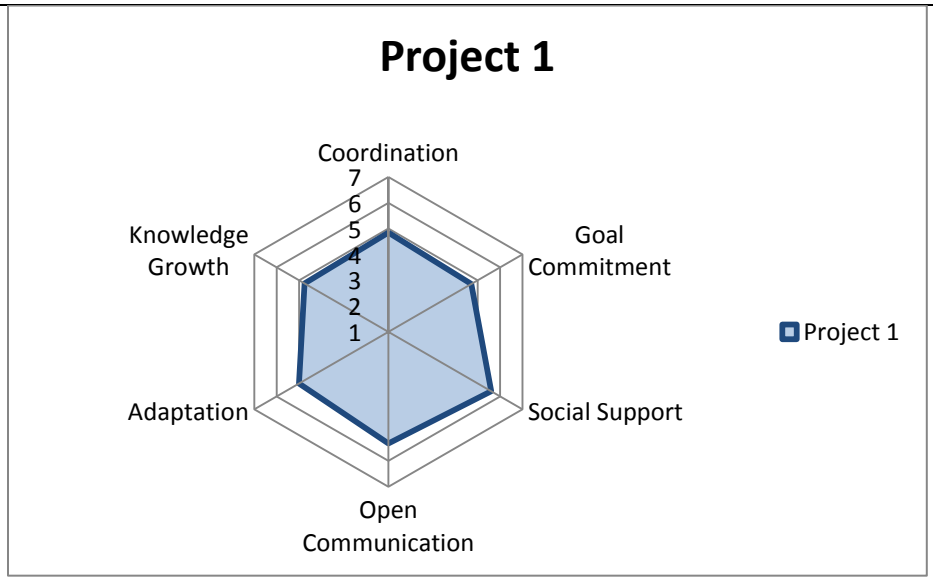
……

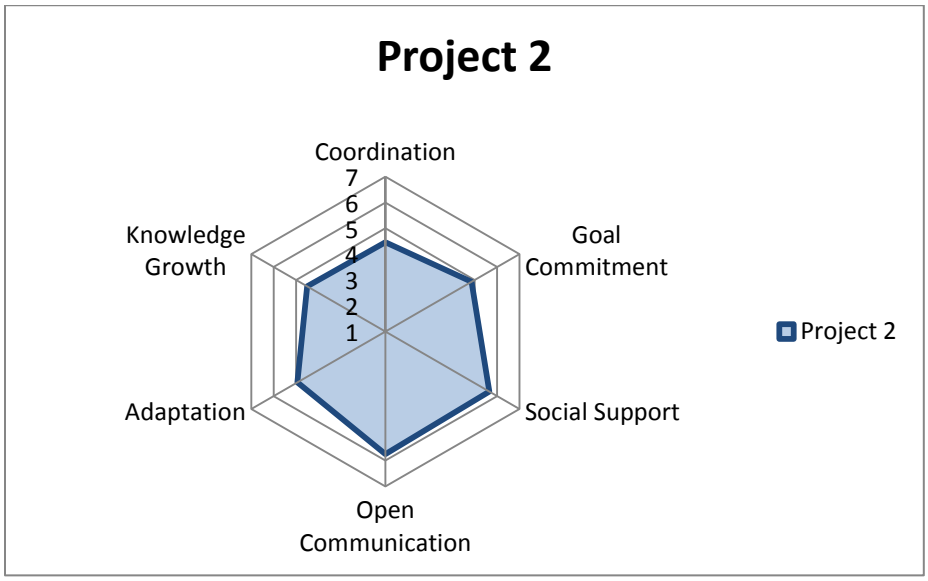
If you are willing to be contacted for possible further questions and/or results of this research, please write down your e-mailaddress here (Optional)

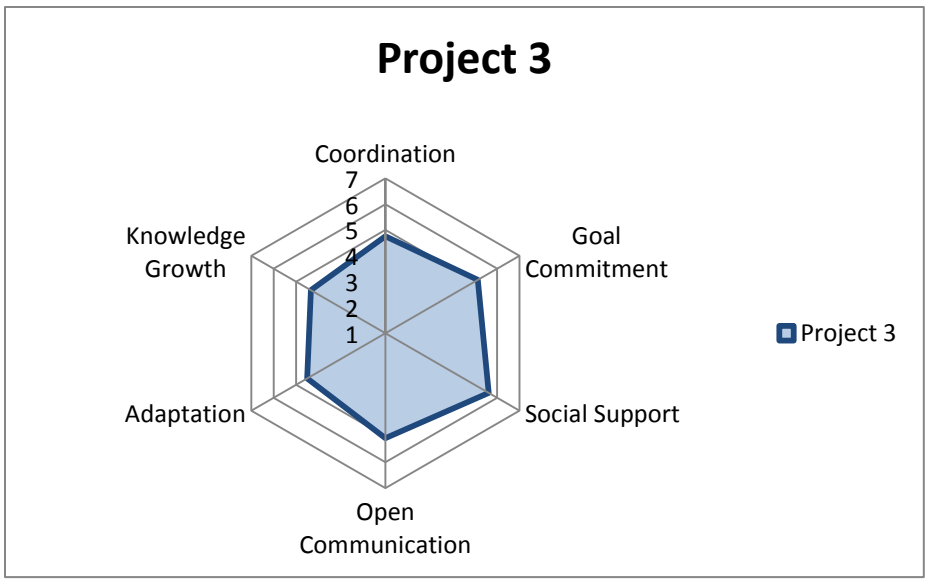
You can leave any additional comments or thoughts here (Optional)

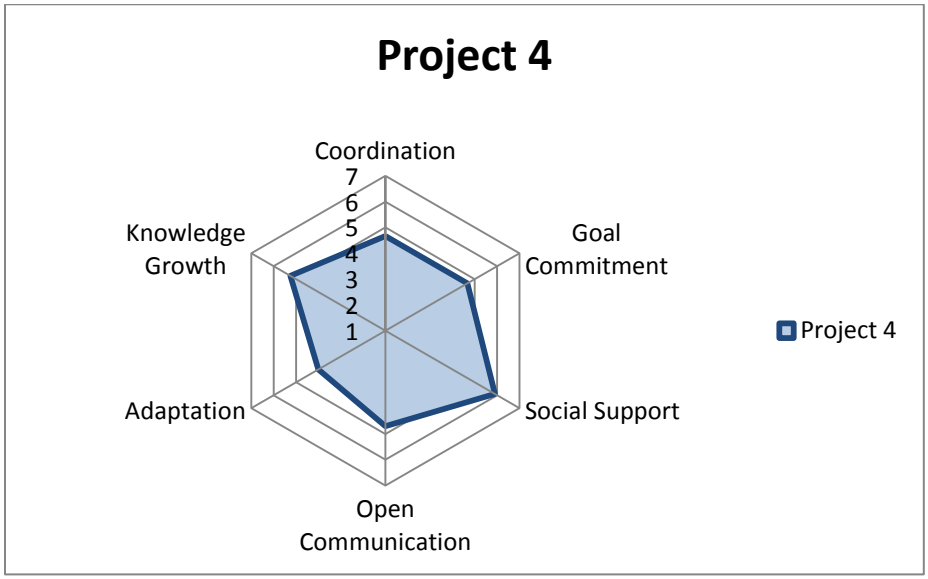
You have reached the end of the questionnaire. Thank you for your participation!

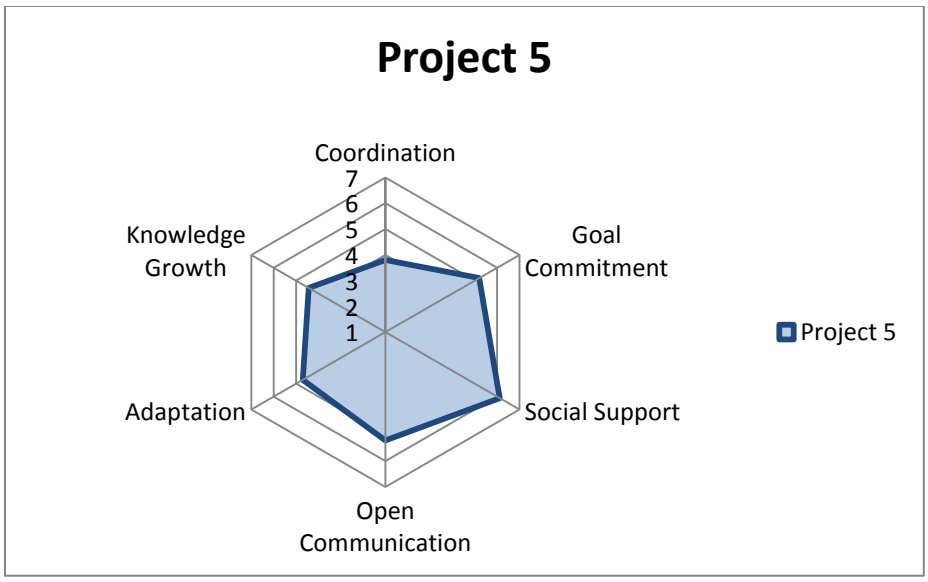
APPENDIX E: SPIDER GRAPHS OF TEAMS

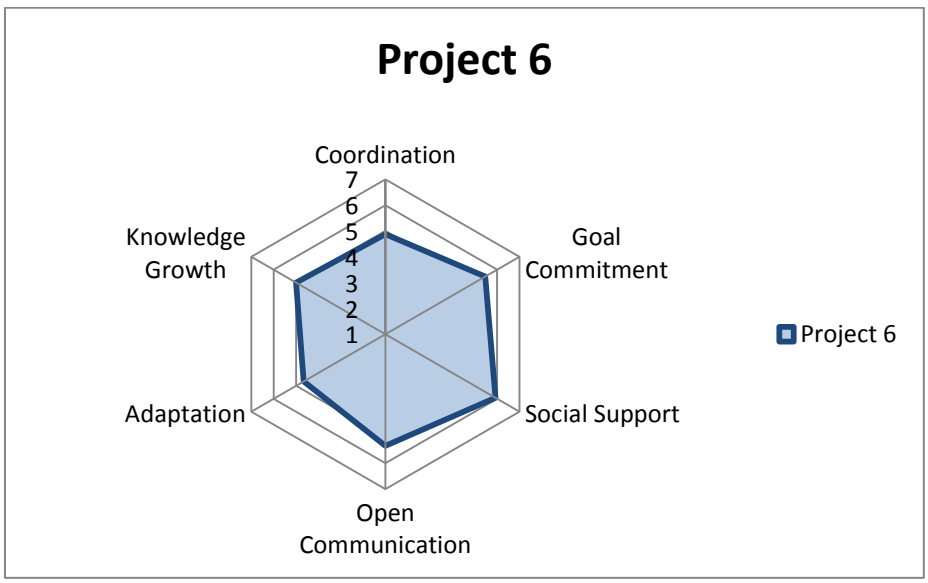


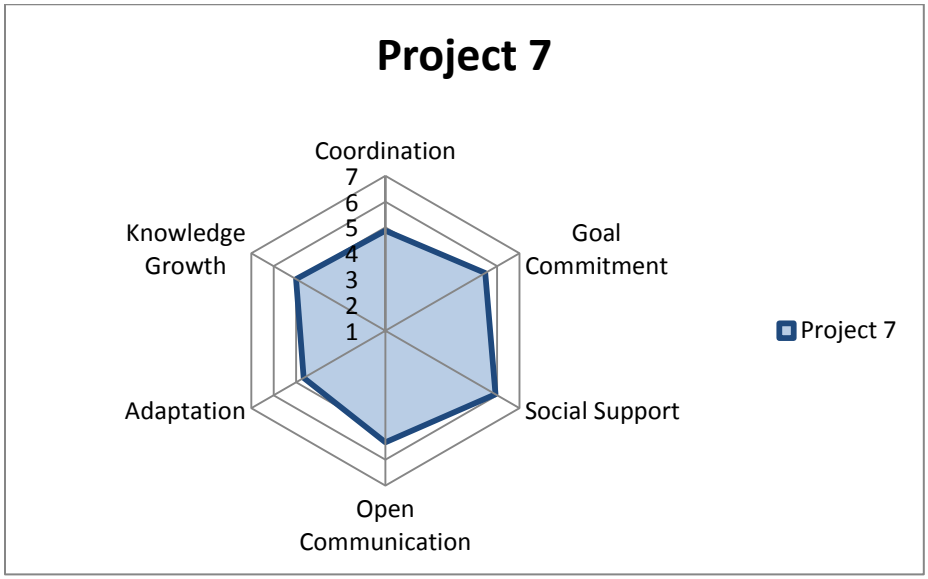


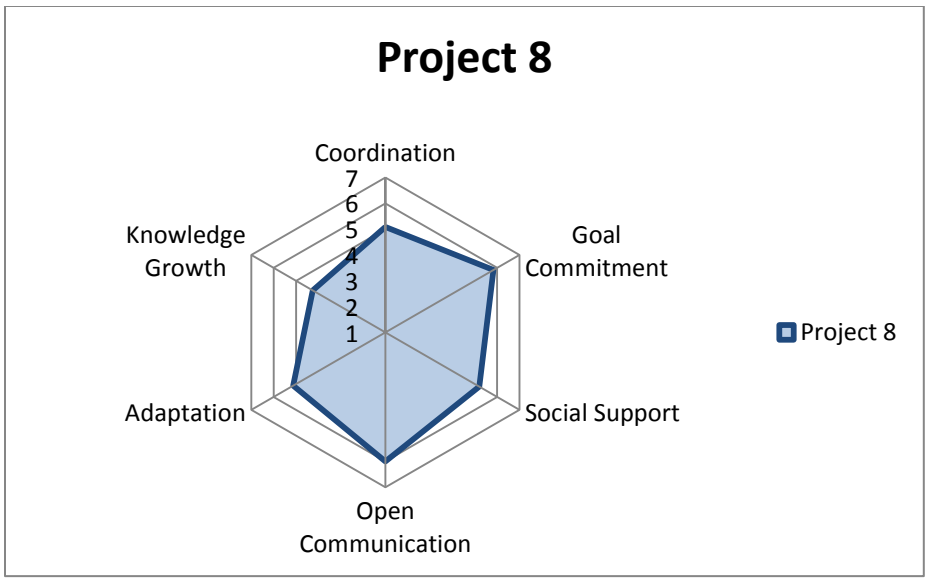


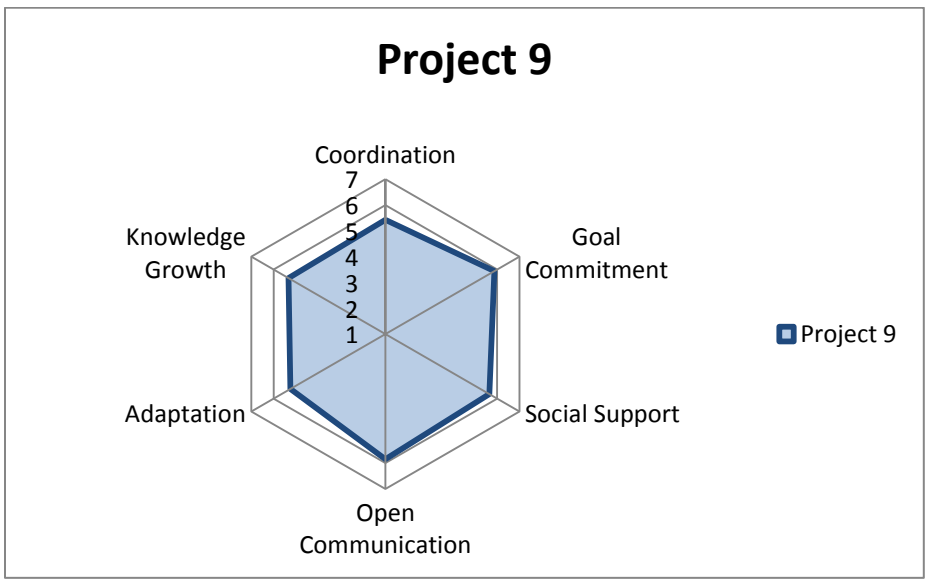

Project 4
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 4


Project 5
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 5


Project 6
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 6


Project 7
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 7


Project 8
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 8


Project 9
Coordination
7
6
5
4
3
2
1
Goal Commitment
Social Support
Open Communication
Adaptation
Knowledge Growth
Project 9

APPENDIX F: ASSOCIATIONS WITH DETERMINANTS FACTORS

Deliver on time	B	Std error	CI 95%
Automated unit testing	-1,140***	0.486	[-2.122 – -0.158]
Prioritized product backlog	1.543***	0.730	[0.064 – 3.021]
Requirements written as informal stories	1.184***	0.558	[0.055 – 2.313]
Deliver on budget	**B**	**Std error**	**CI 95%**
Automated unit testing	-1.190***	0.560	[-2.324 – -0.056]
Customer-visible/valued features	1.580***	0.748	[0.058 – 3.103]
Prioritized product backlog	1.873***	0.815	[0.220 – 3.525]
Deliver high quality	**B**	**Std error**	**CI 95%**
Whole multidisciplinary team with one goal	1.012***	0.495	[0.008 – 2.016]
Prioritized product backlog	1.114***	0.523	[0.055 – 2.174]
Test-driven development acceptance testing	1.026*	0.357	[0.302 – 1.751]
Customer satisfaction	**B**	**Std error**	**CI 95%**
Automated unit testing	-0.771*	0.225	[-1.228 – -0.314]