UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

An Action Research study on the use of Scrum to provide agility in Data Warehouse development

by

Susan Mulder
(22020307)

Submitted in partial fulfillment of the requirements for the degree

M. Com. in Informatics
University of Pretoria
15 November 2010

Acknowledgements

A big thank you to my supervisor, Pieter Joubert, who sat through the tedious task of helping me define my scope when I was not sure what I wanted to do. You played a big part in this, and it was a pleasure working with you

To my parents, whose constant support and love carried me to this point. I could not have done it without you.

To Dawn Taljaard and Prof Carina de Villiers – thank you for placing the faith in me at honors level. I hope that I have repaid it.

Table of Contents

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

List of tables

List of figures

Summary

Data warehousing is a new and emerging field. Projects tend to be complex and time consuming. Because of this complexity, teams tend to commit to more than they can deliver. This causes delayed delivery. Applying Agile development styles to data warehousing is one of the alternative methodologies that are being investigated to help teams to accelerate the delivery of business value.

Scrum is one of the frameworks that falls within the Agile stream. Scrum focuses on project management and makes use of iterative and incremental development. It tries to deliver the smallest piece of business value the fastest. The paper evaluates the implementation of Scrum in a data warehouse team of a financial investment company.

The researcher did an action research study on the team to see if Scrum can be used as a viable alternative framework to bring agility to Data Warehouse development. She examined the changes that the team experienced during and after the implementation of Scrum, focusing on team structure and roles within the teams.

The researcher defined a framework to evaluate how well the team implemented Scrum. The researcher also evaluated the quality of work delivered, and the predictability and productivity of the team as metrics to see if Scrum made a difference within the team.

The research paper examined why the implementation failed and what issues Scrum highlighted within the team as well as within the way that the company implemented it.

Keywords: Traditional software development, Agile, Scrum, Data warehousing, Business Intelligence, Scrum metrics.

1. Introduction

1.1 Background

A data warehouse can be seen as the back-end or infrastructure component used for storing and retrieving the information that a business uses. The term BI refers to the information that is available to a business for decision-making purposes as well as the insights gained from unstructured data and data mining. This information often includes the data stored in the data warehouse (Moss et al., 2010).

Business Intelligence (BI) projects can be very complex, requiring integration from many systems (Moss, 2001). The technology is emergent, and the field is still quite new. With the demand of more data faster, many BI teams make the mistake of committing to more work than they can deliver (Moss, 2001). Applying Agile development styles to data warehousing is currently being flaunted as a means to accelerate the time it takes to deliver business value (Beyer et al, 2010).

The study was partially prompted by the fact that the financial investment company where the researcher works, had decided to implement Scrum in its Web development teams in 2008. Scrum can be defined as “a simple low overhead process for managing and tracking software development” (Clutterbuck et al., 2009). Scrum has a clear emphasis on Agile project management (Clutterbuck et al., 2009). The company wanted to use Scrum to provide agility to the way it was currently developing its data warehouse.

In middle-2008 the company decided to implement Scrum in its BI domain. The domain consists of a reporting team that handles the client facing reporting, and the mart team that is responsible for the maintenance and population of the data warehouse. The data warehouse currently consists of one data mart that is used for reporting purposes.

The researcher joined the mart team just after the initial implementation of Scrum. Although most teams in the company seemed well-adapted to the methodology, the mart team seemed to struggle to adapt. When applying the methodology to the mart team, it highlighted some constraints faced by BI projects that the traditional software development route did not face. These constraints pertained to data validity and a change in the types of unit testing that the domain needed.

The team delivered work in a 2 week cycle. This means that every two weeks they had to have a demonstration to the business of what they had developed. This cycle is called a Sprint in Scrum.

The team developed a history of non-delivery. Over the first 13 sprints, the team failed to gain any consistency or predictability. This can be illustrated by the graph (figure1) below that depicts the delivery of the team .

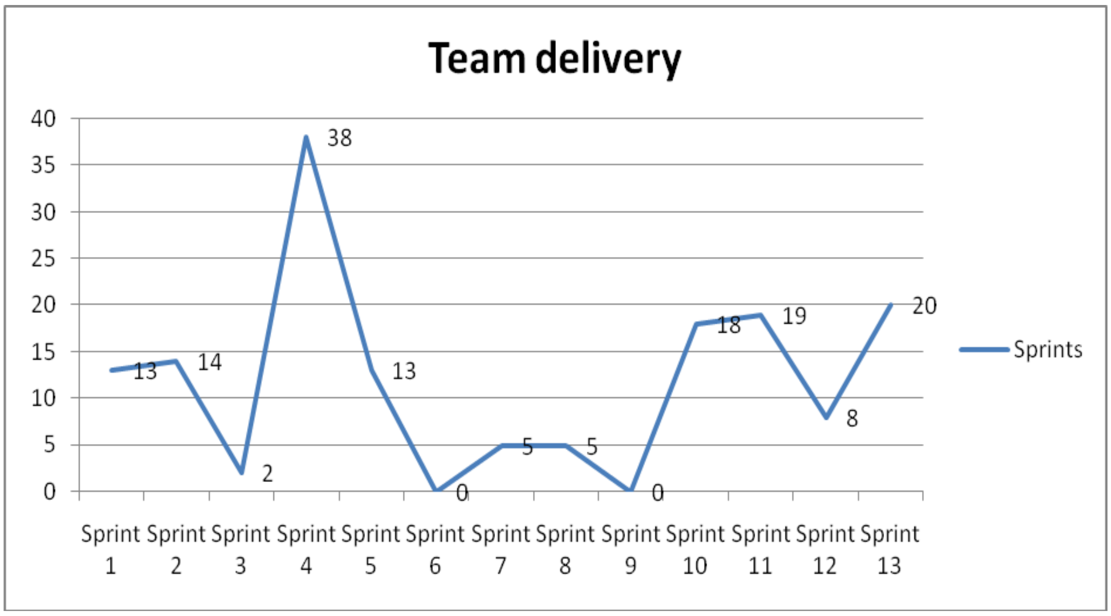


Figure 1Team delivery

The inconsistency in team delivery made the company question the usage of Scrum within the domain. The questions pertained to the way in which it was being implemented as well as the applicability of Scrum to the domain.

In January 2009 (during sprint 13), the researcher became the Scrum Master of the team. As such she became custodian of the Scrum process. The company gave her permission to record the data within the team and to follow the implementation of the methodology. This was done so that she could help the company determine what the issues were and to find ways to solve it.
The aim of the proposed research is to evaluate the effectiveness of Scrum when applied to a non-traditional software development field such as data warehousing. Beyer et al. (2010) stated that the link between Agile and BI is weak. “BI platforms consist of software in an analytics use-case, while Agile is used to develop software” (Beyer et al., 2010:1).

The researcher wants to determine if Scrum can be used to provide agility to data warehouse development and help teams to respond faster and more effectively to changing requirements.

In order to achieve this, the researcher was going to monitor and record the impediments that the team faced, the advantages gained from using the Scrum methodology, and how the team evolved within a defined period from when she took over Scrum process within the team. The researcher will track the team's adaptation of the methodology to suit their needs, their adaption to suit the methodology, and find ways to measure the implementation and effect of Scrum on the team.

1.2 Problem statement

Kimball (2009:1) stated: “In today’s economy the Data Warehouse team is caught between two conflicting pressures. First, we need more immediate, impactful results about our customers and our products and services across the enterprise. In other words, integrate the enterprise’s data NOW! But second, we need to allocate our scarce people and money resources more wisely than ever before. In other words, make sure that all our designs are extensible and universal, and avoid any future rework. We can’t waste a dime!”

Kimball (2009:1) further asked: “Is there a middle ground that we might call an ‘Agile enterprise Data Warehouse?’ Remember that Agile development calls for small teams, a succession of closely spaced deliverables, acceptance of a cut-and-try mentality, delivery of code rather than documentation, and intensive interaction throughout the project from end users who effectively control the project."

This leads to the main research question: Can an Agile project management methodology like Scrum be used to provide agility to data warehousing?

1.3 Research questions and expected results

The problem statement raised several other questions that the researcher will attempt to answer:

1. How is Scrum implemented within a Data Warehouse to provide agility?
2. What are the changes caused by the implementation of Scrum?
3. How can the success or failure of the implementation be measured?
4. Can Scrum provide a balance between traditional and Agile development?
5. How can you measure team performance within Scrum?

The research paper will provide a valid argument for the use of Scrum as a viable alternative framework to bring agility to Data Warehouse development. It will highlight the changes that the team experienced during and after the implementation of Scrum, focusing on team structure and roles within the teams. The researcher will show that Scrum created more generalists in the team, meaning that the tasks are done regardless of the role traditionally associated with the task.

The researcher also defined a framework to evaluate if the team did achieve agility in their data warehouse implementation. This links in with how well the team handled changing requirements, as well as how soon and consistently the team managed to deliver working software. The researcher ensured that the development process that the team followed conformed to all the characteristics of Agile development as described in the literature review.

The researcher will show how the team achieved a balance between Agile and traditional development when it comes to areas in the development life cycle such as analysis. Even though Agile prescribes less documentation, there is still a need for an amount of future analysis and enough documentation to provide a starting point for the implementation. The researcher will show how the analysis can be handled in an Agile environment.

The researcher will examine methods for measuring predictability in the team. It won't be viable for an organization if it can develop software at a greater pace but cannot sustain a constant delivery. This method can be used by Product Owners when it comes to making a commitment to business. The most critical features will have to fall within what the team can confidently deliver, and the nice-to-haves should fall outside of this.

1.4 Limitations of current research

The researcher only focused on the implementation of Scrum within one team and within one company. It will be very difficult to replicate all the factors that played a part in the implementation of Scrum within the above-mentioned context. Therefore it will not be possible to generalize the research results.

There were time constraints as the researcher could only monitor the team for nine months of the year. The research was not sponsored by anyone and was done in the researcher's own time.

1.5 Structure of research paper

The next chapter will provide an analysis of existing literature pertaining to the subject area. Chapter 3 will look at the research methodology that the researcher followed. Chapter 4 will provide a case study on the company and the team that the researcher evaluated. Chapter 5 will describe how the researcher gathered the data used in the research paper, and Chapter 6 will provide a detailed analysis of the data gathered. Chapter 7 will summarize what was learned and draw conclusions.

2. Literature review

2.1 Introduction

The first part of the literature review focuses on the history of software development and then takes a look at the evolution of software development methodologies into the two mainstream methodologies: Traditional software development and Agile software development. It discusses the use of Scrum as an Agile project management framework, traditional data warehouse development, and the use of Agile methodologies in development of the software processes for data warehousing.

2.2 Methodism versus Amethodism

Information system development activities are known as knowledge-work activities that consist of creation, search, discovery, storage and the use of knowledge in an attempt to solve a particular problem. Information system development is mostly guided by a methodology (Meso et al., 2006).

Methodological information systems development is also referred to as teleological software development, or method-ism. It is characterised by two beliefs. The first is that a methodology contains all the suitable information to analyse, specify and design a solution to a problem. The second belief is that systems developers will prefer to work with the methodology if they understand its value (Introna et al., 1997).

Developing new methodologies tends to be driven by technology. It is influenced by the introduction of improved techniques and software tools, as was the case with object-oriented modelling and CASE tools. Some methodologies are developed by blending the strong points of existing methodologies together, and by combining methodologies to compensate for any drawbacks (Nandhakumar et al., 1999).

Methodism is strongly ingrained in information system development. Most literature referring to the history, culture practice and philosophy of information systems development focuses on methodological information system development (Truex et al., 2000).

"The predisposition to believe in the power of methodologies comes from Descartes who proposed that truth is more a matter of proper method than genial insight or divine inspiration" (Hirschheim et al.,1995, as quoted by Truex et al., 2000:57).

Even though different methods require different steps, techniques, sequences and activities, there are common assumptions in all these approaches. One of the major assumptions of methodism is that system developers control and manage the development process. Methods provide the mechanisms to achieve this control (Truex et al., 2000).

Methods fall into three general groups (see table 1):

Type of method	**Description**
Structured	It is a linear model that makes use of clear phases with inputs and outputs in each phase.
Rapid Application Development	It is an iterative model that makes use of high level phases based on some type of prototyping. These methods don't describe in detail the type of techniques that should be used.
Object Oriented	Varies between using an iterative model and a linear model. It provides an object focus for the analysis and design of an information system.

Table 1 Groupings (source: Beynon-Davies et al., 2003)

Information system development methodologies combine procedures, techniques, tools and documentation relevant to the different parts of the system development life cycle. They differ on the techniques and sometimes the whole perspective on the development of information systems (Nandhakumar et al., 1999).

Some of the advantages of information system development methods are that they provide structure to systems development and that they regulate activities. It also reduces redundant activities (Truex et al., 2000). Methodologies are intended to improve the development process, communication of acquired knowledge and productivity of the programming task (Nandhakumar et al., 1999).

Truex et al. (2000:60) stated that the use of methods "permit developers to 'engineer' information flows in an effort to improve the economics of labour and other resources in the production and maintenance of information. Methodologies also enable information systems development to keep up with the demand for new applications, to improve the quality and to reduce the costs of developing these applications (Nandhakumar et al., 1999).

There is a stream of literature that provides a critique of methodologies and their philosophical assumptions. Hirscheim et al. (1989), as quoted by Nandhakumar et al. (1999) identify four ideal types of information system development practises:

1. *Functionalist* – Most mainstream methodologies fall within this paradigm. It is concerned with the technical role of the developer, whose purpose is to convert a set of objectives into a system by using a rational process.
2. *Social relativist* – Fits methodologies such as ETHICS. It sees the developer as a so-called "change-agent who helps to make sense of an emergent world" (Nandhakumar et al., 1999:178).
3. *Radical structuralist* – suggests that "the developer represents labour against capital" (Nandhakumar et al., 1999:178).
4. *Radical humanist* – "concerned with creating conditions of free and open discussion leading to mutual understanding." The discussion also includes a critical examination of existing barriers to emancipation (Nandhakumar et al., 1999:178).

This framework shows the move from seeing information system development as a pure technical process to a more social process (Introna, 1996).

One of the problems with methodologies used in practice is that they are still being used as pure engineering-based methodologies, ignoring or downplaying the social nature of information system development (Introna, 1996). Another argument against the use of methodologies is that the nature of the types of problems faced by software developers has changed in the last decade. The early successes of information systems methodologies lead to extensions of problem domains. But the problem domain has become dominated by problems that are filled with volatility, exceptions, unstructured requirements and unpredictable processing requirements (Truex et al., 2000).

Another view against methodologies is that “they are too mechanistic to be of much use in the detailed, day-to-day organization of systems developer’s activities “(Nandhakumar et al., 1999). Some authors even argue that many of the problems faced by information systems methodologies can be solved by providing users with the tools to write their own applications (Sumner, 1985).

According to Introna (1996) the teleological approach generates systems that are not dynamic enough to meet users’ needs. One solution to this is that information systems development should shift from a system-based schema to narrative, metaphor and myth, as defined by ateleological design.

“If it is believed that information systems are social processes that may, to a greater or lesser extent, be supported by technology, then a teleological approach will be at least limiting and inadequate and at most completely inappropriate” (Introna, 1996:23).

The first principle of ateleological development is that the design process must be in the hands of the people and not a designer, allowing them to design what is useful for them (Introna, 1996).

The difference between methodological and amethodological systems development is illustrated in table 2:

Methodological	**Amethodological**
Information systems development is a controlled process	Information systems development is a random process that is driven by accident. Often opportunistic
It is a linear process, that is executed sequentially	Processes are simultaneous, can overlap and there are gaps
It is a universal process that can be replicated	The development occurs in unique forms
It is a rational, goal-driven process	Information systems development is negotiated.

Table 2 Table Difference between methodological and amethodological systems development (Source: Truex et al., 2000)

Truex et al (2000:56) argued that the concept of using a methodology is so imprinted in our current way of thinking, that information systems development and information systems development method became one concept in system development literature.

The next section describes the two main groupings of methodologies namely traditional software development and Agile software development in greater detail.

2.3 Traditional versus Agile software development

“The optimism that drives method-ism believes that by using method it is possible to understand and solve the typical systems development problem. This leads to the diagram: Method -> understanding” (Introna et al., 1997:33). This mindset thus leads to the design of methodologies in order to understand a problem (Introna et al., 1997).

Introna et al. (1997) suggest a reverse of the above situation. Only through a proper understanding of the context of the problem, will you be able to identify what tools to use. “It is our understanding of what is needed in order to develop a system that makes the tools and techniques emerge as significant. The methodology in itself cannot create or specify this sense of reference or significance. “Once we have an understanding to use the method, we will also understand the limits to methods, which will allow us to use the appropriate methods” (Introna et al., 1997:34).

Traditional software development is also known as the linear sequential model, the "classic life cycle" or the "waterfall model". The cycle proceeds sequentially through phases of analysis, design, coding, testing and support. This is a good model to use when there is a good understanding of the business requirements. If a phase has to be revisited, the process has failed (Gibbs, 1996).

The linear nature of the traditional approach is its biggest problem as it does not define how to respond to an unexpected output from any of the intermediate processes (Schwaber, 1995). Other reasons for the failure of projects that makes use of this approach include requirements that are not fully understood before a project begins, the inability to get proper requirements from users before they see the initial version of the software, changing requirements during the construction process and a change in technology that makes implementation strategies unpredictable (DeGrace and Hulet Stahl (1990) as quoted by Sutherland, 2004).

As a result, 31% of development projects that make use of traditional software development is terminated before it is completed (Sutherland, 2001). Agile software development was developed by experienced practitioners in reaction to the traditional software development methods (Dyba et al., 2008). The Agile philosophy takes the viewpoint that it is better to know after one month that a project is going to fail, rather than waiting 15 months (Brobst et al., 2008).

According to Dyba et al. (2008:835) Agile development should focus on four core values as described in the Agile manifesto:

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

Dyba et al. (2008) identified the following fundamental differences between Agile and traditional software development (see table 3):

	Traditional	**Agile**
Fundamental assumption	Systems are predictable and can be built through extensive planning	Software is developed in small teams using principles of continuous design improvement and testing based on rapid feedback.
Management style	Command and control	Leadership as well as collaboration
Knowledge management	Unequivocal	Implicit
Communication	Formal	Informal
Development model	System development life cycle	Evolutionary development model

	Traditional	**Agile**
Desired organizational structure	Bureaucratic	Flexible and participative
Quality control	Rigid control. Late testing	Continuous testing and control

Table 3 Traditional versus Agile (Source: Dyba et al. (2008))

One of the development streams within the Agile methodologies that can be followed is Scrum. In contrast to traditional software development, it starts with the premise that software development is too unpredictable and complex for future planning. The rationale behind the framework is to provide an approach that will enable development teams to adapt within a changing environment (Schwaber, 1995). The next section will explain Scrum in greater detail.

2.4 Scrum as an Agile methodology

Scrum was developed in 1993 for software development teams at Easel Corporation (Sutherland, 2001). “The name Scrum comes from the English name of a formation of players in the rugby game - a tight formation in which the players are united in a specific position” (Ionel, 2008:435). The approach is based on a Japanese approach to product development where it was implemented in slices until the full product is complete (Sutherland, 2001). Scrum assumes that analysis, design as well as development processes are unpredictable (Schwaber, 1995).

Scrum concentrates on the management of software projects. It makes use of empirical process control mechanisms to ensure visibility, inspection and adaptation. These control mechanisms are implemented to provide flexibility (Schwaber, 1995). Controls of all the different technical and environmental variables are achieved through an iterative and incremental process (Mahnic et al., 2002).

Control is achieved by the use of three principles (Schwaber et al., 2009):

- Transparency – aspects of the process that affects the outcome must be visible to the people who manage those outcomes. It must meet their definition of being done.
- Inspection – the various aspects of the process must be inspected frequently so that variances can be detected. The inspection should take into account all the factors and processes that came into play, and is highly dependent upon the diligence and skill of the people who perform the inspection.
- Adaptation – if one or more aspects within the process fall outside of the acceptable limits, the process must be adapted to minimize the result and to prevent further fallout.

2.4.1 Roles within the Scrum team

The Scrum framework consists of a set of Scrum teams along with their artefacts, rules and time-boxes (Schwaber et al., 2009). Within the team, three distinct roles are defined (Mahnic et al., 2002):

1. The Product Owner represents the interests of business and all the stakeholders in the project. The Product Owner is responsible for compiling and maintaining a prioritized list of project requirements, called a backlog. The Product Owner is also responsible for maximizing the value of the work that is done by the Scrum team (Schwaber et al., 2009).
2. The team is responsible for developing the functionality as required by the business. A Scrum team is self-managing, has to organize themselves, is cross-functional, and has to figure out how to turn the product backlog into deliverable slices of functionality. A team should contain all the skills in order to turn the Product Owner’s requirements into a potentially shippable product (or slice thereof) by the end of a sprint (Schwaber et al., 2009).

3. The Scrum Master takes over the role that a traditional project manager would fulfil, but where a project manager would manage the work, the Scrum Master is responsible for managing the Scrum process. The Scrum Master must ensure that the Scrum process is understood by the team, and followed (Schwaber et al, 2009).

"The goal of Scrum is to deliver as much quality software as possible within a series of short time boxes called 'Sprints' which lasts about a month. Scrum is characterized by short, intensive, daily meetings of every person on a software project" (Sutherland, 2001:5). It allows the developers to choose the techniques they want to use in the development of the software (Abrahamsson et al., 2002).

2.4.2 The phases of Scrum

Scrum is explicitly intended for the management of Agile projects. It was suggested that other approaches be used to compliment the Scrum process, for example, XP (Extreme Programming) (Abrahamsson et al., 2002). The main phases within the framework are indicated by figure 2 (Schwaber, 1995):

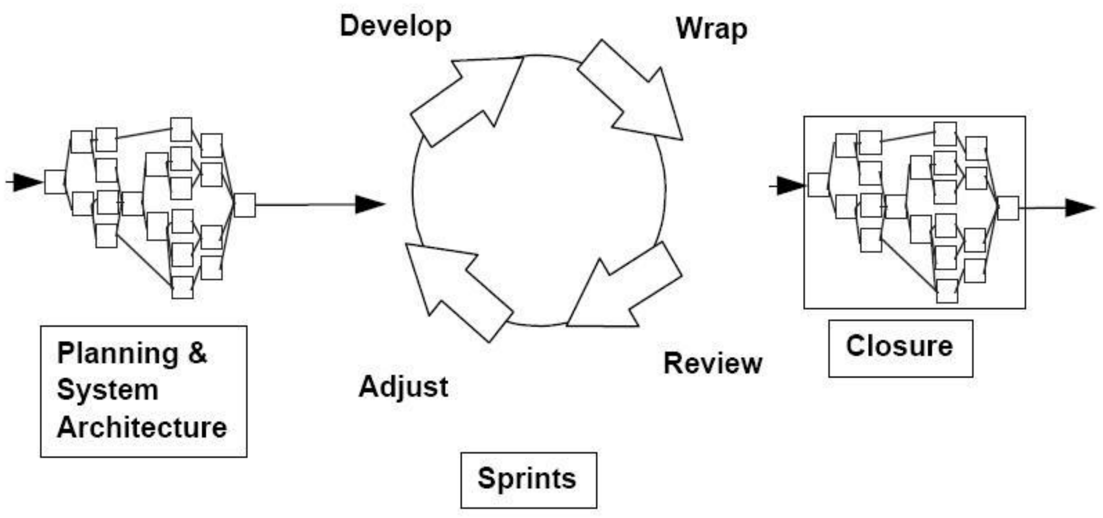


Figure 2 Main phases of Scrum (Source: Schwaber, 1995)

The first and last phases (planning and closure) of the Scrum framework consist of defined processes and the knowledge of how to implement these processes is explicit (Schwaber, 1995). During the planning phase the architecture for the project is defined (Rising et al, 2000).

The sprint phase is empirical, containing many processes that are not identified and not controlled. This phase is treated as a black box, and it requires external controls. These controls are implemented within the sprint iterations in order to maximise flexibility (Schwaber, 1995).

Sprints deliver the functionality. A sprint is an iteration of a month or less. It is consistent length throughout the development process. Sprints deliver increments of the final product (Schwaber et al, 2009). If applicable, explicit process knowledge is used during the sprints, alternatively trial-and-error is used to build process knowledge (Schwaber, 1995).

Deliverables can be changed any time before the closure phase of the process. This allows the project to adapt to environmental complexity throughout all the phases (Schwaber, 1995). The closing phase involves delivery and that the team reflects and improves (Rising et al., 2000).

Scrum makes use of time boxes in order to implement regularity within a project. Elements that are time-boxed include the sprint planning meeting, the sprint itself, the daily Scrum, the sprint review and retrospective, and the release planning meeting (Schwaber et al, 2009). These meetings fall in the planning and closure phases of the Scrum framework.

2.4.3 The release planning meeting

The purpose of the release planning meeting is to establish a plan and goals of a release. Risks are evaluated, delivery dates are determined and the overall features and functionality that should be developed in the release are established. This can be inspected and changed on a sprint-by-sprint basis. Release planning requires the estimation and prioritization of the backlog for the release (Schwaber et al, 2009).

Advantages of the release planning meeting includes allowing individual milestones to be aligned with the overall project goal. It also... “[a]llows different methodologies for various project estimations, Facilitates risk management and contingency planning within next Sprint and Customer Designed Features Testing aligns business functionality with software correctness – the right product for customer.” (Clutterbuck et al., 2009:21).

2.4.4 The sprint

Changes that affect the goal of a sprint should not be allowed during the sprint. A sprint consists of a sprint planning meeting, the actual development, a review and a retrospective. Sprints should occur with no time between the sprints (Schwaber et al, 2009).

A sprint may be cancelled before the allocated time for a sprint is over, but only under instruction from the Product Owner. This may be done because of changes in the market or technology, or if a sprint no longer make sense in the given circumstances. Cancellation of a sprint is very rare because of the short time span of a sprint. Upon cancellation, all completed stories are reviewed, and the rest are placed back on the product backlog (Schwaber et al, 2009).

2.4.5 Sprint planning meeting

The sprint planning meeting is used to plan an iteration. The meeting should be time-boxed to approximately 5% of the total sprint length and consists of two parts, sprint planning 1 and sprint planning 2 (Schwaber et al, 2009). The sprint planning meeting is hosted by the Scrum Master (Wake, 2004).

During sprint planning 1, the team addresses what should be built. The Product Owner presents a prioritized backlog for the team. The team then selects the highest priority items on the backlog, building a sprint backlog. After that the team defines a sprint goal. At the end of this meeting, the team has to output a sprint goal as well as a print backlog (Wake, 2004).

The second part is used by the team to design the product and to define tasks for the sprint backlog that was defined in the first meeting. The team designs the product and defines tasks during this session. Tasks are detailed pieces of work that can be completed in less than a day that will convert the product backlog into software (Schwaber et al, 2009).

In order to complete the work as defined by the meeting, the team needs to self-organize without outside influence. In order to achieve this, the team must work together and realize that the whole team, and not individuals, is responsible for delivery (Schwaber et al, 2009).

Clutterbuck et al. (2009) identified that a few of the advantages of this meeting include that trust between the customer and the developers is built sooner, and risks are identified and managed a lot earlier. The involvement of both customers and business analysts in the testing of functionality leads to more accurate specifications with fewer changes in subsequent iterations.

Some of the risks are that the customer does not know enough about the processes or that he/she does not have the authority to take decisions. There might be a disconnection between the project manager and the customer. The technical ability of the business analyst can also play a role in the decisions that are taken (Clutterbuck et al., 2009).

2.4.6 The daily Scrum

The daily Scrum is hosted by the Scrum Master and is used to answer three questions:

- What did you do yesterday?
- What will you do today?
- What will keep you from doing it?

It should occur the same time every day, and during this session the team updates the sprint backlog (Wake, 2004).

2.4.7 The sprint review

The sprint review meeting is held at the end of a sprint and should not take up more than 5% of the total sprint time. This a collaborative session between the team and the stakeholders on what has been done (Schwaber et al, 2009). It is hosted by the Scrum Master and should be attended by everyone who was involved (Wake, 2004).

The Product Owner identifies what has been completed and what has not been completed. The team demonstrated the work that has been completed and should be able to answer any questions regarding the work that has been done (Schwaber et al, 2009). It is an open discussion between the team and the stakeholders as they determine if it meets all the requirements (Wake, 2004).

The team can comment on what problems they ran into during the development and how they fixed those problems. They can also discuss what worked well for them. The Product Owner discusses the current product backlog with the team as well as likely project completion dates. The sprint review should provide input prior to the first sprint planning session (Schwaber et al, 2009).

The review is useful because it allows the confirmation of quality (on a technical as well as a business process accuracy level) of each component as it is completed (Clutterbuck et al., 2009).

2.4.8 The sprint retrospective

The retrospective is the last meeting and fits between the review and the next sprint planning meeting. The retrospective inspects how well the development went in terms of relationship, process and tools. The team evaluates what went well and what can be improved (Schwaber et al, 2009).

2.4.9 Scrum artefacts

The artefacts used by the Scrum framework are summarized in table 4 below:

Artefact	**Explanation**
Product Backlog	A list of requirements ordered by highest priority of what is needed
Sprint Backlog	List of requirements and tasks needed turn the backlog for 1 sprint into a product
Release Burndown	A graph that measures the product backlog over time for a release
Sprint Burndown	A graph that measures sprint backlog items across the time for a sprint

Table 4 Artefacts of the Scrum process (Source: Schwaber et al, 2009)

During the course of the sprint and the release the team will produce and manage these artefacts. The backlogs are used to maintain work that still needs to be done on a sprint and release basis, while the burndown tracks what work has been done and when it was completed (Schwaber et al, 2009).

Schwaber (1995) compared the characteristics of Scrum to other methodologies (see table 5):

	Waterfall	**Spiral**	**Iterative**	**Scrum**
Defined processes	Required	Required	Required	Planning and closure only
Final product	Determined during planning	Determined during planning	Set during project	Set during project
Project cost	Determined during planning	Partially variable	Set during project	Set during project
Completion date	Determined during planning	Partially variable	Set during project	Set during project
Responsiveness to environment	Planning only	Planning primarily	At end of each iteration	Throughout
Team flexibility, creativity	Limited – cookbook approach	Limited – cookbook approach	Limited – cookbook approach	Unlimited during iterations
Knowledge transfer	Training prior to project	Training prior to project	Training prior to project	Teamwork during project
Probability of success	Low	Medium Low	Medium	High

Table 5 Characteristics of Scrum compared to other methodologies (Source: Schwaber, 1995)

The Scrum methodology has the following benefits (Ionel, 2008):

- It is flexible along the project life, and allows organizations to modify the project and deliverables at any time
- Scrum teams share knowledge in a good learning environment

Rising et al. (2000:8) also commented on the following end result of Scrum: “As a result of the interaction of small teams in small, focused development cycles:

- the product becomes a series of manageable chunks,
- progress is made, even when requirements are not stable
- everything is visible to everyone,
- team communication improves,
- the team shares successes along the way,
- customers see on-time delivery of increments,
- customers obtain frequent feedback on how the product actually works
- a relationship with the customer develops, trust builds, and knowledge grows, and a culture is created where everyone expects the project to succeed.”

One potential weakness of Scrum is the availability of external clients who have to test changes on periodical bases. Close interaction might not be possible (Ionel, 2008). Another shortcoming is that integration and acceptance tests are not specified within Scrum (Abrahamsson et al., 2002).

The researcher would have liked to examine articles on the use of Scrum in non-traditional fields such as BI since this is the focus of the research being conducted, but no academic articles could be found to support the use of this methodology in the development of a data warehouse. Instead, the next section will focus on data warehousing and the use of Agile methodologies within data warehousing, lastly highlighting the possible applicability of Scrum as an alternative Agile methodology.

2.5 Traditional versus Agile data warehousing

The primary goal of a data warehouse is to provide access to information that can be used for decision making (Brobst et al., 2001). It offers access to integrated and historic data from varied sources. The value of a data warehouse is derived from the use of the data in the data warehouse (List et al., 2002). Data warehousing is the result of business need and technological advances (Wixom et al., 2001).

Data warehousing is a young discipline compared to software engineering and does not yet offer the established techniques and strategies for the development process. The next section will give an overview of the different traditional data warehouse development methodologies, as well as the Agile alternatives.

2.5.1 Traditional data warehousing development methodologies

Current methods can be divided into the following groups: data-driven, goal-driven and user-driven (List et al., 2002).

2.5.1.1 Data-driven methodologies

Advocates of the data-driven methodology argue that data warehouse environments are data driven and not requirements driven. Requirements are only understood after the data warehouse has been populated and users had the opportunity to analyze the results of their queries. The needs of the users as well as company goals are requirements that are only integrated in a second cycle (List et al., 2002).

2.5.1.2 Goal-driven methodologies

The goal-driven methodology is based on the semantic object model (SOM) process modelling technique in order to derive the structure of the data warehouse. The first stage determines goals as well as services that the company delivers. The second stage analyzes the business processes by applying the SOM interaction schema. The third step transforms sequences of transactions into sequences of dependencies and the last step identifies your measures and dimensions from these dependencies (List et al., 2002).

This approach is quite complex and only works well when the business processes are aligned with business goals. An alternative is the four-step approach, where a business process is chosen, takes the grain of the process and then chooses the relevant dimensions and facts to fit that process (List et al., 2002). In this context, a business process can be defined as “a major operational process in the organisation that is supported by some kind of legacy system (or systems)” (List et al., 2002:3).

2.5.1.3 User-driven methodologies

The user driven methodology was developed by Wal-Mart and focuses on implementing business strategy. It is based on the assumption that the company goal is the same for everyone.

A prototype is set up that reflects the needs of the business. Goals and business questions are defined and prioritized by the business. Top priority business questions are then defined in terms of data elements. This approach focuses on business needs, but business goals as defined by a business are ignored (List et al., 2002).

2.5.1.4 Triple- driven methodology

There is a fourth methodology called triple-driven methodology that combines all three of the above-mentioned methodologies. It consists of four stages (Guo et al, 2006):

1. Goal-driven stage – produce subjects and key performance indicators (KPIs) of the main business fields

2. Data-driven stage – obtains the data schema

3. User-driver stage – produces the business questions along with the analytical requirements

4. Combination stage – combines the above-mentioned result.

Some of the advantages of this methodology include (Guo et al., 2006):

- The data warehouse reflects the long term-strategic goals of the business, ensuring business value and stability
- It promotes trust and acceptance from the users
- It ensures that the data warehouse can support a wide range of analysis
- It leads to a comprehensive design

2.5.2 Problems with data warehousing

One of the problems with data warehousing is that it is hard to determine what data will provide the value before working with the actual data (Brobst et al., 2001). Many data warehousing projects fail because of the complexity of the development process (List et al., 2002). Watson et al., as quoted by Wixom et al., 2001 listed weak sponsorship, weak management support, politics within the organization, poor user involvement and insufficient funding as the most common failure reasons. Gardner (1998) summarized five primary reasons why data warehouses fail:

1. Lack of partnership between the IT department and the business users
2. Incorrect data warehouse architecture
3. Experience levels of the team
4. Inadequate planning
5. Dependency on technology

Factors that also add to the complexity of data warehousing applications are large scoped dirty data, and few standards or common methods. Thus BI applications cannot be developed without some kind of rigour (Moss, 2001). Aside from the above-mentioned complexity, each BI project can actually be defined as three parallel software development tracks since this includes (Moss, 2001):

1. The back-end extract-transform-load (ETL) – responsible for design and populating the databases
2. Front-end applications (data access and analysis) – reports, queries and views
3. Metadata repository – added information about business data

Moss (2001) defines the following development steps in the engineering of a data warehouse (see table 6):

Step	**Explanation**
Justification	An assessment of the business problem or opportunity
Planning	Includes strategic and tactical plans, and how the project will be accomplished
Business analysis	Detailed analysis of the business problem, providing an understanding of the business requirements
Design	Data warehouse is conceptualized
Construction	Data warehouse is built
Deployment	The finished product is implemented and evaluated to determine whether it meets the expectation

Table 6 Steps in engineering a data warehouse (Source: Moss, 2006)

With the maturity of the data warehousing industry, there came a greater demand for agility (Brobst et al., 2008).

2.5.3 Agile data warehouse development methodologies

The Agile methodology values individuals, software and collaboration highly. The philosophy that drives these values in data warehousing makes satisfying the end-user requirements through the timely and continuous delivery of analytic capability the highest priority. There is a greater emphasis on delivery than on the process that is followed (Brobst et al., 2008).

The aim of Agile data warehousing is to “create a culture of innovation wherein data exploration and construction of new analytic application is encouraged without the overhead of long project cycles” (Brobst et al., 2008:2).

Hughes (2008) gave the following reasons to emphasize the need for an Agile data warehousing methodology:

1. Traditional software development data warehouse projects yield notoriously poor return on investment.
2. Customers are getting increasingly impatient to get tangible results for a data warehouse instead of a “big bang” approach that is developed over a long time.
3. Business cannot avoid tackling the unknown. Data warehouse teams do not know many of the characteristics of the software they are going to build at the start of a project.
4. Infrastructure problems cause problems late in the development cycle.
5. Business rules become clear very late in the development cycle.
6. Documentation for future work has a very short lifespan because of the changing nature of data warehousing.

There are different ways in which to achieve agility in data warehouse development. This next section will discuss three of these methods, namely Integrated sandboxing, Extreme scoping, and Agile data warehousing.

2.5.3.1 Integrated sandboxing

One option is to use integrated sandboxing as an Agile framework. The idea behind an integrated sandbox implementation is to provide a way for the developers to get the data into an environment without having to develop a full-blown methodology. The sandbox is a special database area that will be used without any of the formal controls that are defined. Raw data will be loaded with minimal cleaning and will allow the data to be examined much sooner (Brobst et al., 2008).

The proposed Agile framework can be measured for success by measuring the rate at which data becomes available to knowledge workers (Brobst et al., 2008). One of the core advantages of using this approach is that “it provides an incremental method of delivering new content to knowledge workers without encouraging the proliferation of data marts” (Brobst et al., 2008).

2.5.3.2 Extreme scoping

Moss (2001) proposed the use of extreme scoping as another alternative. The aim of the method is to reduce the complexity of a project in order to achieve a balance between rigour and agility. The deliverable equates to a fraction of what would typically be requested.

Extreme scoping is based on the same concepts as prototyping, but it is not the same as traditional prototyping. Where traditional prototyping is ad hoc development, extreme scoping includes all the activities appropriate to the scope of each release with the rigour of a methodology. The scope is no longer the entire application, thus reducing the complexity of the application (Moss, 2001).

The problem with this approach is that many users do not realize that the project will be developed as an incremental approach, and that more functionality and more data will follow with each subsequent release (Moss, 2001).

There are some concerns regarding the use of Agile in BI projects like data warehousing. Moss (2001) went as far as comparing the use of Agile without rigour to “the thrills of chaos”. She commented that Agile and rigour should be used in conjunction in order to move away from the so-called chaos.

The problem with obtaining agility in data warehousing is the complexity of the projects. These projects are usually developed across multiple systems causing them to have many cross-organizational interrelationships. Added to this complexity are the large scopes that are associated with BI projects (Moss, 2001).

Moss (2001) reasons that because of the complexity and the risks associated with it, the complexity must be minimized in order to obtain agility. It is a known fact that most BI teams over-commit during a time frame (Moss, 2001). Their scope is too big for the development time that they are given.

Moss (2001) argues that methods and guidelines should be created to merge rigour with agility. Hughes (2008) proposed a method called Agile data warehousing (ADW).

2.5.3.3 Agile data warehousing

Agile data warehousing is defined as “the application of two Agile development approaches – Scrum and Extreme programming – to the specific challenges of Data Warehousing and BI” (Hughes, 2008).

Hughes (2008) defined a six-step process in order to implement Agile data warehousing. He suggests that the team start with a generic Scrum implementation, and then they can specialize in subsequent stages. The six stages that Hughes (2008) defined are generic Scrum, user epic decomposition, formulating the release plan, reference models and test-led development, pipelined delivery squads, and automated and continuous integration testing.

Stage 0: Generic Scrum

This stage has three objectives:

1. Team members should sit together in a shared workspace.
2. The customer should be embedded within the team as the Product Owner.
3. Team members can organize their work in any way they please as long as they can transform a significant part of the Product Owner’s backlog into a shippable product in two to four weeks.

Hughes (2008) also suggests that no formal project manager is needed. A team member should be trained as a Scrum Master and be responsible for guiding the team forward. The team is introduced to the Scrum terminology of user stories, task boards and burndown charts. These tools will help the team to establish a velocity and form the starting point for consistent estimation of stories.

The team should work in the environment for a few sprints, with the Product Owner reviewing and approving the work at the end of an iteration (Hughes, 2008). This step provides the foundation for the next five steps. The team can move on to the next stage once at least 60% of the story points is accepted by the Product Owner for two consecutive sprints and once the following benchmarks are achieved (Hughes, 2008):

- Team members work comfortably with the story vocabulary for defining requirements
- The team can smoothly progress easily through all the required meetings of Scrum (planning session, task break down, user demo and retrospective)
- Daily Scrums are completed within 15 minutes
- Basic infrastructure is in place for the project
- Team can estimate by story points
- Team can create do-able task cards
- The burndown chart is reliably updated

Stage 1: User epic decomposition

Once the team is comfortable with the workings on Scrum, they can begin on honing their working habits for data warehousing (Hughes, 2008). The purpose of this stage is to increase the team's velocity and to improve estimation accuracy up to a point where the Product Owner regularly accepts at least 80% of the story points (Hughes, 2008).

At this stage, the team introduces the Product Owner to the BI-specific architectural notions so that the team acquires a common vocabulary. This ensures that all the details of a BI application are addressed in order (Hughes, 2008). The team should produce user stories conforming to the “INVEST” qualities of stories, namely small, independent, testable and estimable.

Along with the standardization of vocabulary, the team has to learn to estimate their work based on the relative size of requirements. The team can begin to derive standard estimates for standard units of functionality. The team should identify a set of reference stories to which they compare other stories in order to achieve relative-size estimation. This will increase the team's understanding of the number of requirements it can deliver within a sprint (Hughes, 2008).

The team can move on to the next stage when the Product Owner accepts 80% of stories for two or more consecutive sprints. The Product Owner should have enough information to properly prioritize stories and to assure external stakeholders of delivery times and costs (Hughes, 2008).

Stage 2: Formulating the release plan

Hughes (2008) suggests that at this stage the team has gathered enough knowledge on both the system as well as their velocity to know how fast they can create the components. The team can now start to work on a release plan. In the plan, the team captures the full scope of the desired release and the number of iterations it will take to deliver the remainder of the data warehouse.

The objective of Stage 2 is to publish a release date and the release plan. The release plan is a list of major features that will be delivered, grouped into sprints. The stage concludes with a recommitment from business for the project, as well as funding for the project. The team can only proceed once the project sponsor grants the team the authority to proceed based upon the plan it provided (Hughes, 2008).

Stage 3: Reference models and test-led development

Once permission to proceed is obtained, the team can start to focus on the quality of the modules it is producing. This stage focuses on the guidelines that the developers will use to engineer their software. Reference models keep the number of formal specifications to a minimum. Test-led development forces a degree of quality into the code. Providing a set of re-usable validation objects will save labour without losing days on short-lived to-be specifications (Hughes, 2008).

The developers have two objectives in this stage, namely compiling the reference model and defining tests for the modules before coding them (Hughes, 2008). The criteria for a sufficiently defined reference model are reflected in its usage by the team members (Hughes, 2008):

- "Listing the tasks required to deliver a newly started module
- Specifying the features required to make it 'bullet-proof' and restartable
- Estimating the level-of-effort required
- Design-by-exception, that is, stating that an upcoming module will behave and be coded like a unit from the reference model except for a certain list of differences
- Documentation-by-exception, that is, drafting prose that indicate it matches the reference unit set except for a particular set of differences"

Additional objectives for this stage can also include (Hughes, 2008):

- "Patterns identified for design and coding
- Complete basis-of-estimates cards for units in the reference model
- 2-1 design sessions for the modules to be coded and later for peer reviewing the units when they are complete
- Module validation scripts written before modules are coded, that is, scripts that can automatically apply the tests defined for a module
- Self-documenting code"

The focus on quality will cause an initial drop in velocity. As the team reaches their quality objectives, they can refocus on regaining velocity. Stage 3 is complete when the team is comfortable with the defined quality standards and when 70% of the stories are once again accepted by the Product Owner (Hughes, 2008).

Stage 4: Pipelined delivery squads

The team should now be delivering modules that are more robust than before at a slower pace. In order to combat the stress experienced by the developers because of a high workload, this stage of the methodology guides the team to re-organize into semi-formal squads where each team has a specialization, for example design, coding and integration.

By organizing these squads into a progressive pipeline, each time has a full iteration to give its full attention to quality. By keeping the assignments semi-formal, the teams have the ability to move labour in order to facilitate delivery of the most important deliverables (Hughes, 2008). After a few iterations, the team should see an increase in their velocity.

Stage 4 is successful when the following conditions emerge:

- Members move between squads as needed so that there is now downstream delay
- Emerging modules still possess all the required quality aspects
- The velocity of the development squad should reach at least 80% of the velocity at the end of stage 2.

Stage 5: Implement the automated testing facility

The final stage of the methodology establishes an integration environment where each module is integrated once it is defined as done. By continuously validating a candidate release using predefined data sets, errors are detected fast and efficiently (Hughes, 2008).

Stage 5 is complete once the following criteria are met:

- The integration squad can display the artefacts created by automated testing during the demos to prove that the code is shippable
- The team can promote the current code to production in accordance with the wishes of the Product Owner.

Hughes (2008) states the following advantages of implementing Agile data warehousing:

- It is a method with specific deliverables and a predictable, drill-down sequencing and supports any level of maturity that an organization desires
- It ensures the continuation of IT services
- It supports common architectures
- It provides budgeting and progress metrics
- It offers visibility and the correction of underperformance
- It ensures the quality of the developed software
- It enhances IT departmental cohesion

There are some risks to implementing Agile data warehousing. Team members cannot easily be redeployed. Added or removing new members causes a change in process and team velocity, making up-to-date budgeting and planning forecasts unreliable.

The business must wait longer for budgeting and planning estimates. It will take Agile teams a few months before they can start giving dependable estimates as to when features will be completed. This is because Agile teams wait until they know the project before giving estimates (Hughes, 2008).

Another risk is that the Product Owner has to be a part of the team. If the Product Owner is not available, it undermines the team’s ability to function accurately. Agile often projects the full focus of their Product Owners because they cannot keep them fully occupied (Hughes, 2008).

2.6 Gaps in current literature

In examining articles regarding Scrum, the researcher found very few articles on the Scrum process as well as little empirical evidence of its use. Schwaber and Beedle (2002) as quoted by Dingsoyr et al. (2010:48) claim that "[Scrum] practices have been established through thousands of Scrum projects". But none of these projects were cited, which invites scepticism.

More recently, however, a study can be found where Scrum was tested in real-life projects, thus showing that the body of needed industrial evidence is gradually growing, and that Scrum has some empirical support (Rising et al., 2000).

Another gap is that many of the research papers found were on the theory of the use of methodologies in the development of Information Systems, but few articles focused on the application of the methodology in practice (Sutherland, 2001). There seems to be a lack of academic articles that include the viewpoint of practitioners.

The researcher could find very few academic articles on the usage of Scrum or any other Agile methodologies in the development of a data warehouse, and no empirical proof of its effectiveness in this area. The researcher hopes that her thesis will make a contribution to fill this gap.

3. Research methodology

3.1 Background

The researcher’s close involvement with the Scrum team to be studied will not allow the researcher to be neutral. She will be an active participant within the team and not an impartial observer. This will imply that the data that the researcher collects will be interpreted based on her bias and opinions. Therefore a positivist approach cannot be followed.

3.2 Philosophical position

Interpretivism tries to understand the social context of an information system (Oats, 2006). Clutterbuck et al. (2009) said that the interpretive study was particularly suited in new and evolving fields. Thus it would fit in perfectly with the usage of a project management technique like Scrum in a non-traditional software development field like data warehousing. There is little to no research available on the above-mentioned usage.

Some of the characteristics of interpretivism include (Oats, 2006):

1. There are multiple subjective realities. This means that there is more than 1 version of the truth.
2. Meaning is constructed dynamically and socially. Language and shared meanings are different for different groups.
3. The researcher is reflexive and therefore not neutral.

The above-mentioned characteristics make interpretivism the ideal viewpoint for the researcher, as she is closely involved in the team as both a researcher and a team member. She cannot distance herself from the research as she is helping to construct meaning and interacting within the team. The research that will be conducted is exploratory in nature, as the researcher does not have preconceived notions.

The researcher will only be focusing on one case study. Where positivism is criticised by using a single case study, it is encouraged in interpretivism (Clutterbuck et al., 2009). The aim is not to generalize, but to monitor the team and to implement within the specified context.

Oats (2006) defined the criteria for interpretive research as follows (see Table 6):

Criteria	**Explanation**
Trustworthiness	The research is evaluated not in terms of validity, but in terms of trustworthiness.
Conformability	Another researcher must be able to trace through the data and find the conclusions that the researcher had drawn.
Dependability	The research should be well documented and traceable.
Credibility	Was the research conducted in such a way that it can be proved as credible?
Transferability	Can the findings of one case be transferred to another case?

Table 7 Criteria for interpretive research (Source: Oats, 2006)

The researcher had to ensure that she met the criteria mentioned above. This held the following implications for her research:

- The researcher was part of the research setting and as such she had to be careful not to let her own bias affect the results and information. In order to do this, the researcher collected as much information as possible, and all the findings were collaborated by the data.
- She documented retrospectives, the decisions made and how it was implemented within the team. This provided a basis to prove that the researcher did indeed conduct action research within an interpretivist domain.
- The researcher aimed to document how Scrum can provide agility to data warehousing. Her findings on her topics included a description of the implementation that other teams can use if they want to try and implement Scrum in the same context.

The aim of all interpretive research is to understand how members of a social group, through their participation in social processes, enact their particular realities and endow them with meaning, and to show how these meanings, beliefs and intentions of the members help to constitute their social action. The interpretive perspective attempts “[to] understand the intersubjective meanings embedded in social life” (Gibbons, 1987 as quoted by Orlikowski et al., 1991:13).

The research will provide a first-hand account of how the team went about to implement Scrum in a data warehouse environment and aims to document and note its impact on the team and how it changed the way that the team develop their software and reach their commitments.

3.3 Research approach

The researcher used action research as her chosen research approach. Action research has been defined as a “post-positivist social scientific research method, ideally suited to the study of technology in its human context" (Baskerville et al., 1996, p. 235).

“Action research aims to solve current practical problems while expanding scientific knowledge. It is a clinical method that puts IS researchers in a helping role with practitioners” (Baskerville et al., 2004:329). It is the responsibility of the action researcher to assist practitioners in both the development and application of knowledge (Lindgren et al., 2004).

Action research is one avenue to help improve the practical relevance of Information Systems research. “Action Research has been accepted as a valid research method in other applied sciences such as Organization development and organization” (Baskerville et al., 2004:329).

Action research is an iterative process that capitalizes on the learning of both the researcher and the subjects within their social system. While studying the organization, the researcher is also concerned with creating change in the organization (Baskerville et al., 2004).

Action research consists of a two-phase process. The first phase, called the diagnostics phase, analyses the social context of the subjects and formulates theories about the research domain. The second phase, called the therapeutic phase, is concerned with bringing in change and studying the effect of the changes (Baskerville et al., 2004).

"Action researchers bring knowledge of action research and general theories, while clients bring situated, practical knowledge" (Baskerville et al., 2004:330).

3.3.1 Premises of action research

Action research is based on four premises (see table 7):

Premise	**Explanation**
Consequences define human concepts	• Based on an argument by Charles S Pierce namely that in order to focus on a particular concept, one needs to determine the purpose and consequences of that concept • "In order to experiment, or to even determine purpose, people must have independence of will to decide what actions they could choose to undertake" (Baskerville et al., 2004:331) • Focus on ideas and concepts

Premise	**Explanation**
Practical outcome embodies truth	• Originates from William James, who stated that "The search for truth, and the continuous reinforcement of truth, was centralized in the mind" (Baskerville et al., 2004). • Focuses on truth • "Human thought can only be revealed in human action. Practice is central in discovering and revealing this truth" (Baskerville et al., 2004)
Logic of controlled inquiry	• Based on the idea of John Dewey • Focus on education and learning • "Discovered a common five-element pattern to all forms of human inquiry: (1) An indeterminate situation, (2) formulation of a problem, (3) determination of a solution, (4) reasoning, (5) operationalization of facts" (Baskerville et al., 2004:332) • Dewey argues that reasoning alone cannot force change, and that action, guided by reasoning, is the only way to reorder a particular setting
Social context of action	• Defined by George Herbert Mead • "Any acting individual will respond to his or her own acts to some degree as other individuals respond. Our social consciousness is a reflection of ourselves mirrored in the reactions of others. A social act makes one a social self, a self-observer and responder" (Baskerville et al., 2004:332) • Emphasis on the need for collaborative teams

Table 8 Premises of action research (source: Baskerville et al., 2004)

3.3.2 How action research is conducted

Action research is conducted following the premises as stated above. It can be summarized in the following four steps (Baskerville et al., 2004):

1. Establish the purpose of any action
2. Find the practical action in the problem setting
3. The theory is adjusted according to the outcome of the practical action
4. The action researchers participate in the research while observing the results as well. The research has to be conducted within a social setting

There are two forms of action research, namely Canonical action research and Dialogical action research. The next section will discuss these in detail and explain what method the researcher decided to use.

3.3.3 Different forms of action research

Canonical action research is known for its rigorous, iterative and collaborative process. It can be defined in the following five steps (Lindgren et al., 2004):

1. Diagnosing – identifying problems
2. Action planning – specifying what actions to take
3. Action taking – implementing the actions specified in step 2
4. Evaluating – assessing the result of step 3. This is performed by the researcher and the practitioners
5. Specifying learning – the documentation of the ongoing process

In dialogical action research, the researcher accepts the practitioner as the expert of the system, and attempts to communicate with the practitioner on the same level (Martensson et al., 2004). Through one-on-one dialogue that happens in a setting outside of the practitioner’s offices, the researcher and the practitioner build a mutual understanding. The researcher suggests an action based on scientific theories for the practitioner to take (Martensson et al., 2004).

3.3.4 Advantages of using action research

The advantages of action research can be summarized as follows:

- It brings theory and practise closer together (Lindgren et al., 2004)
- It is one of the few methods geared towards change (Baskerville et al, 1999)
- Action research provides a balance between rigour and relevance (Martensson et al., 2004).
- Dialogical action research allows for the researcher and the practitioner to communicate on the same level (Martensson et al., 2004).
- “It is problem-focused, context-specific, and future-oriented.” (French, 2009:190)
- “It helps to develop a holistic understanding.” (French, 2009:190)
- “It can use a variety of data collection methods that suit an organisation’s environment” (Sankaran and Tay (2003) as quoted by French, 2009:190).

3.3.5 Critiques of action research

The disadvantages and critiques against the use of action research can be summarized as the following:

- Issues about what defined action research (Baskerville et al., 1998)
- Validity issues, including the credibility of the researcher (Baskerville et al., 1998)
- Little attention has been devoted to the exact processes by which such theories are cyclically developed during the course of action research (Baskerville et al., 1999)
- “The problem with action research arises from the fact that it cannot be wholly planned and directed down particular paths... The researcher may express his research aims as hopes, but cannot with certainty design them into his ‘experiments’. He has to be prepared to act to whatever happens in the research situation; he has to follow wherever the situation leads him or stop the research” (Checkland (1981) as quoted by Baskerville et al., 1999:5).

Iversen et al. (2004) identified the following pitfalls:

- Researcher may be partial
- There can be a lack of discipline
- It may be confused with consulting
- Because of the context, it might be hard to generalize findings

3.3.6 Establishing the validity of action research

Canonical action research can be evaluated according to the following five principles (Lindgren et al., 2004):

- The principle of researcher-client agreement given the importance of collaboration in action research, this principle seeks to ensure that researchers and practitioners (clients) develop a mutual understanding of and commitment to the research project, i.e., its scope, focus and mode of inquiry.
- The principle of the cyclical process model: this principle highlights the importance of rigour in that it advocates progressing through all five action research phases in a sequential and systematic manner.
- The principle of theory: seeing that action research without theory does not constitute research, this principle highlights the importance of using one or more theories to not only guide and focus the research activity, but also relate the findings to the extant literature.
- The principle of change through action: since the purpose of action research is to change an unsatisfactory situation, this principle stipulates that interventions appropriate to the problem and the client organization should be designed and implemented.
- The principle of learning through reflection: this principle highlights the importance of drawing insights from the research and identifying implications for other situations and research contexts.

The researcher will be conducting canonical action research on the proposed research problem. The next chapter will discuss the team and company structure that the researcher used as a case study.

4. Case study

The researcher worked at a financial investment company. As mentioned in chapter 1, the data mart team that the researcher observed maintains and builds a reporting data warehouse for a financial investment company. The main purpose of the data warehouse is to provide data for any client facing reports as well as for the company's website.

The customers that the mart team have to supply with data are all internal to the company. The clients are typically business people wanting ad hoc data extracts, the web development team want access to client information or the reporting team that has to generate reports for the business.

The next sections will give a historic overview of how the team implemented and adapted to Scrum, as well as some of the changes to the team structure and the process that the team follows in the development of the processes. The impact of the mentioned changes will be discussed as part of the data analysis chapter.

4.1 Implementing Scrum within the data warehouse team

The company has always followed an Agile development approach, but started to implement Scrum in March 2008 within its web domain. Subsequently it moved the implementation into the other domains, with the BI domain being the last team to implement Scrum in 2008.

The company has fixed two-month release cycles. The first part of the cycle is used for the development of new software, and the last two weeks of the cycle is used for cross domain testing and integration, as well as for user testing.

4.1.1 Team structure

The BI team started Scrum in September 2008. At the start of the implementation the team consisted of a system analyst, three business analysts, three developers and a Product Owner. The lead business analyst was appointed as the Scrum Master for the team, performing a dual role.

In November the decision was made to split the team into two teams. This was done because the team got into a situation where some of the developers had to wait for development on the data warehouse to be done before they could start writing the reports to consume the data. The newly formed mart team consisted of one system analyst, two business analysts and two developers. The lead business analyst still acted as Scrum Master for the team. A second team, called the reporting team, was formed with the remaining people as well as two new developers. The decision was made that the Product Owner would manage two backlogs for each of the teams.

In December 2008 the mart team lost its system’s analyst. The duties of the systems analyst were absorbed within the team. The four-member team was now responsible for making the design and architecture decisions.

In March 2009 the researcher took over the role of Scrum Master that was previously held by the business analyst. During this time the company hired a Scrum coach to evaluate the team and to try and solve some problems within the team. The observations from the Scrum coach can be found under Appendix C.

In June of 2009 the team’s roles and responsibilities shifted. Some of the analysts started to develop, and the developers had to become more involved in testing data. The traditional roles started to disappear and the team moved to a more generalist approach. This caused friction between team members. Management had to run an intervention within the team, and an external Scrum coach was hired again to evaluate the team and to help them adjust to the changes.

At present the team is structured in such a way that any team member can perform any tasks, even though the team members still have their task preferences.

4.1.2 The development process

The team does development on two areas of the mart. The staging side of the data mart contains all the transformation tables that result from the manipulation of the data into acceptable standards. The statement side contains the final product and is used by all the consumers.

When developing any new functionality for the data warehouse, there are five areas that the team has to focus on. These are design, schema changes to the data mart, process changes, testing and deployment.

4.1.2.1 Design

Once the team receives the requirements for the new functionality, they decide on the table structure as well as the business rules that accompany the data. Initially the system analyst used to do the upfront design with the developers implementing it and the analysts doing the testing.

With the changed team structure, the team was now responsible for the design of the data warehouse. The business analysts would propose a design, and then the team discusses the implications of the design as well as alternatives until everyone is happy with the proposed solution.

4.1.2.2 Schema changes

All the tables for the mart reside on an SQL server. Once a design has been accepted, the team creates or alters the relevant schemas on the data mart. Initially the team wrote each script by hand.

At present the team uses a spreadsheet to generate the scripts for them. These scripts are then numbered according to the order that they need to be run in, and stored as part of the deployment.

4.1.2.3 Process changes

The team uses SSIS (SQL Server Integration Services) packages as well as SQL scripts to transform the data and to get it into the data mart. The ETL process can be broken down into its three sub-processes, namely extract, transform and load. The transform and load process will always get impacted when new functionality is added, and the extract will only change if the new functionality requires data that is not currently being extracted from the source systems.

There are different types of dimensions and facts within the data mart. A type 1 dimension is updated or appended to, a type 2 dimension stores a history of changes, and type 3 dimensions is a hybrid of the first two, where only certain columns hold a history of changes.

Extract

The extract process is responsible for pulling data from the source systems onto a staging database. New data that must be extracted will result in a new SQL script that is added to the extract package. After a change is made, this script as well as a new copy of the extract package must be added to the deployment.

Transform

The transform process links all the packages together that are responsible for populating all the different facts and dimensions on the database.

Initially the analysts would provide the developers with a proposed data dictionary for the process. The developers would then consult with the analysts in order to define a source query and implement the process. The analysts would then test the data once the process populated the table, and provide feedback to the developers. This process would continue until the analyst was satisfied, and then the code would be propagated up to the next environment just before a release.

When Scrum was implemented, the team changed the process. Instead of the analysts just doing the analysis and providing the team with a data dictionary, the analysts started to develop the source queries, and provided the developers with a source script instead. The developers then implemented the package and the analysts would once again do the testing.

As the process evolved, the analysts progressed to developing packages. This caused some bottlenecks in the team as all of the knowledge was situated with the analysts. The team changed the development process again, including the developers in the analyses.

The team sits together to come up with a design for the new requirements. Once the team has defined the design, they define the required schema and identify what data they need from the source systems. Lastly they set about transforming the data into an acceptable form for the data mart.

The team first defines a source query to bring the data in, and then team members define all the lookups to other dimensions in order to derive any keys or additional information that has to be stored in the mart. Once the data is transformed, it is stored in a temporary table and the data in the staging table is then updated or new data is inserted based on the type of the table.

Load

Initially all the data was loaded into dimensions on the staging area, and then the database objects were copied over to that statement mart. As the team evolved their development process, they removed the copy object script. The load process was modified to load data directly from the transformation tables on the staging side into the set of tables on the statement side.

4.1.2.4 Testing

The team did not have any test suite available for testing. The analysts each had their own scripts for data testing on the different tables.

As part of the testing process, the team created an automated testing solution. Test cases are defined and added to the solution, where it would then be executed, with the results written into a table. The team then goes through the results and clears any failed test cases.

4.1.2.5 Deployment

When it came to deploying the code and schema changes to other environments, the team used to rely on the operation section to run each script manually. In order to get the sequence and numbering of the scripts correct, the team used a spreadsheet to organize and generate the deployment sequence.

The next step was to create a package that would automatically run all the scripts. The team developed a numbering system for the scripts, and then manually added each script to the deployment.

The team also implemented SMS notifications to notify the members if something went wrong with the deployment. In an attempt to further automate the process, a team member developed an application that would automatically generate the deployment packages without the team touching them. At present the team only needs to execute one package in order to deploy their code to another environment.

Along with evolving their processes, the team also had to evolve the way in which they implemented Scrum.

4.1.3 The Scrum implementation within the team

The following section will describe how the company went about introducing the team to Scrum, as well as how the process was implemented within the team.

4.1.3.1 Introducing the team to Scrum

The company hired a Scrum coach to provide two-day training to the team. They were introduced to all the concepts and artefacts of Scrum, a Scrum Master was chosen, and the team started their first sprint on 20 August 2008.

4.1.3.2 Estimation and velocity

The team uses a technique called planning poker in order to estimate their stories. The team uses the Fibonacci sequence to estimate stories in terms of effort, rather than hours. In order to form valid estimations, the team had to establish a baseline estimate. The team would then compare different stories based on the effort given to that story, and give it an estimate based on that.

The estimates are used to give a relative idea of the size of a piece of work. The more work the team thinks it will be, the higher the effort estimate. The team initially struggled with this concept. A story that they estimated as an effort of 13, and that the team committed to completing within two weeks took them six months to complete. The team restarted their estimates twice, defining a new baseline and estimating on its relative size.

During the last restart, the company got an external Scrum coach in again to evaluate and work with the team in order to help them with their estimates and Scrum implementation. With his help the team picked one story and examined it in terms of effort, and then they compared all the other stories in their backlog to that story in order to determine effort. The team currently still uses that baseline in their estimations.

The team struggled to reach a predictable velocity. The following graph depicts their velocity:

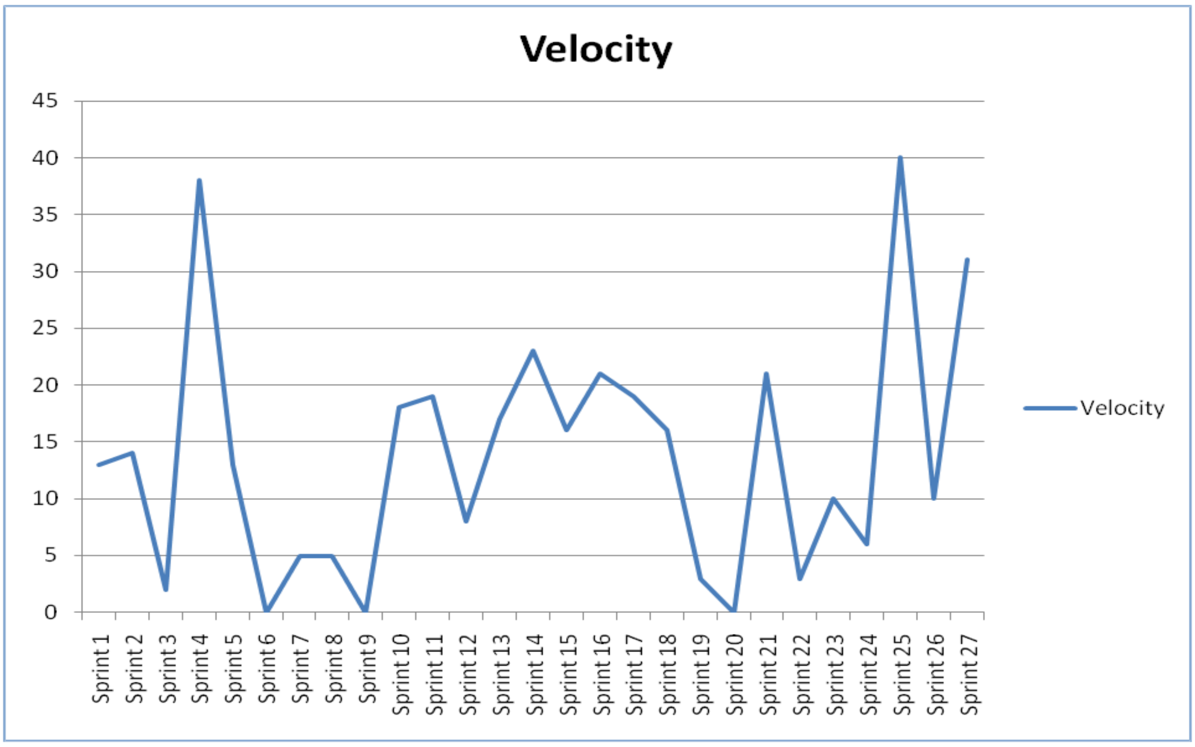


Figure *3 Velocity*

This graph will be discussed in greater detail in the data analysis chapter.

4.1.3.3 Sprints

The team started off by following a two-week sprint cycle. Sprint planning 1 and 2 were held on a Tuesday or Wednesday morning. The team followed a two-week cycle for 10 months before converting to a threeweek cycle.

4.1.3.4 Scrum meetings

As part of their implementation, the team had the following six meetings: the daily Scrum, sprint planning 1 and 2, backlog estimation sessions, the review and a retrospective. This will be discussed in more detail in the next section.

Daily Scrum

The team conducts the daily Scrum meeting every morning at 9:30. It is attended by all four team members as well as the Product Owner. The team first states what they did the previous day, and then moves on to what they will be doing for the rest of the day.

Initially the meeting lasted almost 40 minutes. The session turned into a reporting and problem-solving session, and the team had a tendency to have side conversations not related to the work in progress. It got to a stage where team members were skipped or did not bother to take their turn. The team focused their attention on the meetings and brought it down to ten minutes.

Sprint planning 1

The sprint planning 1 meeting is time-boxed to an hour-and-a-half. In sprint planning 1, the team goes through the backlog with the Product Owner. They discuss what is required for each story and evaluate whether the estimate that they gave the story is still valid.

Sprint planning 2

The sprint planning 2 meeting is also time-boxed to an hour-and-a-half. The team uses this meeting to come up with a high level design for the stories that they have committed to deliver. They also identify what tasks they will need to perform in order to complete a story. All of these were written up and placed on a big task board in the team area, visible to the whole team.

Backlog estimation sessions

The team did not have any backlog estimation sessions. They only estimated stories once they got into sprint planning 1. In September 2009 this changed, and the team now has a backlog estimation session once a week for an hour. During this session, the team discusses and estimates upcoming stories, and tries to determine what questions they still have before they commit to delivering the story.

Review

During the hour-long review, the team shows the Product Owner the new schema that they created. At present, the team still struggles to find a way to actually show a product to the Product Owner, as the actual business value lies in the reports that are pulled from the data. These reports are not created by the team, though.

Retrospective

The retrospective is scheduled for two hours at the end of each sprint. The team first goes through a time line to identify events that they can recall from the past sprint. It can be personal or work-related.

After the time line, the team identifies things that worked well for them in the past two weeks, as well as things that didn't work or that they would like to improve on. The team members then each gets a vote as to what items they would like to fix in the next sprint, when the team discusses options for it and implements it as an action in the next sprint.

4.1.3.5 Stories and task definition

Stories were initially written in technical terms instead of in business value terms. When the Scrum coach came to work with the team in September, he helped the Product Owner and the team to write stories in terms of the business value that they would deliver. All the stories now follow the pattern: "As a _____ I want to ____ so that ____"

Typically, a story would read: "As an investor I want to be able to see a summary of all my account information so that I can determine how much money is invested in what account".

4.2 Motivation for using action research in an interpretive context

The set-up of the team at present was perfect for action research because of the following factors:

1. The company was already using reflection and improvement. Software was developed in two-week increments, and released every three months. After every two-week increment, the team got together to reflect upon what had happened in the past two weeks, and how they could improve on it. This gave the researcher the opportunity to make small improvements and to monitor the effects of these improvements.
2. There was a problem in the team. Since the implementation of Scrum in August 2008, the team had not delivered. There had been role changes, people removed from the team, even threats were issued, but despite it all, the company was not able to pinpoint why the team did not deliver.
3. The researcher was already part of the team. She moved to the team at the end of August 2008 as a junior developer. The team knew her, and they worked well together. She had been at the company long enough to know the process as well as the dynamics of the team that she was to evaluate.

She was made Scrum Master in the beginning of March (this person is the facilitator of the Scrum process, removing any impediments that the team might face, etc.), so this places her in the perfect position to perform the action research.

4. In conducting action research, the team can inform themselves as well as the researcher. The team is supposed to be self-managing, and this will add to their body of knowledge.
5. This research will be valuable to the researcher and the company, and because of that, she already had the cooperation of all the key stake holders.

The research was based on events and situations that happened in the past. The researcher ran an intervention every two weeks, allowing the team to reflect on their implementation, what they did well and what they did wrong. This allowed the team to change the way they were doing things.

Action research is the best approach because of the researcher’s close involvement within the team. It helped her maintain the balance between being a researcher and being a practitioner. Based on the experience level of the researcher, it helped her to draw from the experience of the other practitioners within the team. Because the software development cycle is only two weeks, it helped her in the rigour of information collection and data generation.

It was important that the researcher should not let her own bias cloud her interpretation of the data. In order to ensure the validity of the data collected, the researcher provided the reader with a proper audit trail of how she recorded the data and how she came to her conclusions.

4.3 Ethical considerations

When conducting research within the company, the researcher will have to keep the following ethical considerations in mind:

- The identity of the company must be kept confidential, as it is the wish of the company not to be identified.
- The researcher should not let her own bias towards situations influence her findings or the conclusions that she draws. This will particularly come into play if the researcher examines team structure and the relations within the team that is being evaluated and other teams.

5. Data collection

The data collection chapter will detail how the researcher conducted the research and how the research data was generated.

The researcher studied a BI team within a financial company that she worked for. The researcher was part of the team for six months prior to starting the research, but only evaluated the team during the period that she was Scrum Master of the team. The date range when the team was evaluated was from 6 March 2009 until 3 November 2009. During this time, the team completed 14 sprints. For more detail on the case study, please refer back to Chapter 4.

The company releases new functionality to its users on a one-and-a-half month cycle. During the time frame of March to November 2009 the company had four releases as summarized in table 8:

Release	**Date**
Release 15	March 2009
Release 16	May 2009
Release 17	August 2009
Release 18	October 2009

Table 9 Release Summary

5.1 Research method

According to Cresswell (1994) as quoted by Kalermo et al. (2002:112) “the qualitative study approach is considered an appropriate method when little is known about the phenomenon under investigation and the concepts are immature due to lack of theory and previous research and a need exists to explore and describe the phenomena”.

The goal of the research was to investigate and monitor the usage of Scrum in a non-traditional software development field. The researcher could find no academic studies or articles on the implementation of Scrum in a data warehouse and as such the concept is new and not yet widely studied in the academic world. Thus a qualitative approach seemed appropriate for the study.

The researcher decided to conduct action research on the team. The general action research methodology is described in Chapter 4. This section will describe how the researcher conducted the action research.

Action research is conducted following these four steps (Baskerville et al., 2004):

1. Establish the purpose of any action.
2. Find the practical action in the problem setting.
3. The theory is adjusted according to the outcome of the practical action.
4. The action researcher is participating in the research while observing the results as well. The research has to be conducted within a social setting.

The implementation of these steps is described in the next sections, as well as the data generation techniques.

5.1.1 Research design and procedure

The retrospective of each sprint was the perfect platform for action research. It is attended by the whole team and the aim of this session is to improve the process within the team. It is an open floor where the team can discuss any issues that they have. In the retrospective the team evaluates their performance during the previous sprint. The retrospectives were facilitated by the Scrum Master, who was responsible for recording all the relevant information.

During the first part of the retrospective the team looks at what went well during the past sprint. All these ideas are then written down and grouped together. In the second part of the retrospective, the team writes down anything that they were not satisfied with or any item that they think the team can improve on. After all the ideas are written down and grouped together, each member gets a number of votes to place against the items that they feel are the most important. Each member had two votes.

The items are then ranked according to votes, and the team decides how many of the items they want to take into the next sprint in an attempt to fix it. They then define actions for that item, and also designate people to take responsibility for it.

The researcher recorded all the actions listed during these sessions, and will use it to determine how many of the items the team ended up implementing and improving on.

The researcher needed to find a way to measure the implementation of Scrum within the organization. One way of measuring whether a team is doing the basics is by doing the Nokia test (Sutherland, 1999). It is a ten-question test that evaluates the following areas: iterations, testing within the sprint, Agile specifications, the Product Owner, the product backlog, estimates, sprint burndown, team disruption and the team.

The researcher decided that the test was too high-level and not specific enough for the purpose of the study. She wanted a deeper focus on all of the actions that should happen within Scrum, including all the meetings, not just the iteration and people involved in it. She also wanted to evaluate how well each role was fulfilled by the team.

As a result the researcher designed her own framework in order to determine if the team followed the Scrum process. After conducting a literature study on the Scrum framework, the researcher determined four main areas that the team would be evaluated on namely artefacts, meetings, roles and process. (For the full framework, refer to Appendix A.)

Artefacts refer to all the deliverables that fall within the Scrum framework. It consists of the backlog, the sprint burndown and the release backlog.

Meetings refer to all the meetings that should happen on a daily, weekly or monthly basis. It includes sprint planning 1, sprint planning 2, the review, the retrospective, the daily Scrum and the estimation meetings.

The next section evaluated the **roles** within the Scrum team and how diligent people were at fulfilling the responsibilities for that assigned role. The subsections looked at the Product Owner, the Scrum Master and the team.

The last section, **Process**, evaluated all the actions that should happen per sprint, as well as impediments.

For each of the sections the respondent had to answer a number of questions. The questions were compiled from literature on what people do wrong within Scrum, and had four possible answers: never, sometimes, mostly and always. The questions were structured in a positive manner, i.e. the answer would be *Always* if the team was doing it correctly.

Each of these answers has a point value associated to it. The following table reflect the point value:

Answer	**Point value**
Never	0
Sometimes	1
Mostly	2
Always	3

Table 10 Point value assigned to answers

The framework was drawn up in a spreadsheet, and these values were automatically calculated. The researcher decided that in order for the implementation to be successful, the benchmark would be 80%.

The researcher needed to find a way to measure the impact that Scrum has on the team. She needed some metrics that can provide evidence that Scrum is working or not working within the team. Beherns (2008) defines Metrics across four dimensions when it comes to evaluating a team in Scrum: productivity, quality, predictability and value.

The different metrics is summarized as follows (see table 11):

Dimension **Metric**	**Description**
Productivity Sprint cycle time	Measure the turnover of stories by when it starts versus when it finishes. A short cycle of time is a good indicator.
Sprint work in process	Measure the amount of work in progress. A lot of work in progress is a bad indicator.
Sprint completion bar	Measure points against done, not tested, not coded and waste as a bar graph. Short done bars are a bad indicator as well as any waste. Long not tested and not coded bars are also a bad indicator.
Completed versus Committed % (a.k.a. earned value)	Measures the completed work versus the original commitment expressed as a percentage. This metric was identified as a myth because teams can manipulate this metric if they know that they are being measured.
Quality Technical debt point	Measures how much technical debt is incurred to determine the quality of a product. A high number is a poor indicator.
Running automated tests	Measures the number of tests that pass with each sprint. A low pass rate is a poor indicator.
Post sprint defect arrival	Measure the number of defects that arrive after the sprint where the code was developed.
Post release defect arrival	Measure the number of defects that is found after code was released to production.
Root causes fixed	Measure the number of defects that identified and fixed a root cause.

Dimension **Metric**	**Description**
Predictability Enhanced burndown chart	Measure velocity and scope change throughout a release. This is done as a measure of meeting a particular goal set for the release.
Velocity	Measure velocity change in a release.
Cost per story point	Determine the average cost per story point.
Hours per story point	Measure the average number of hours estimated per story point.
Value metrics Business value delivered	Assign business value to each item delivered per sprint, and measure the business value delivered per sprint.
Customer satisfaction survey	Get qualitative and/or quantitative feedback from customers.
Employee satisfaction survey	Get quantitative and qualitative feedback from employees through a survey.

Table 11Summary of Scrum metrics (source: Beherns, 2008)

The researcher decided to use one metric per dimension in order to measure Scrum within the team. The metrics used were:

- Productivity - completed versus committed % (a.k.a. earned value)
- Quality - post release defect arrival
- Predictability - velocity
- Value metrics - customer satisfaction survey

The researched used the burnt down charts to gather data for velocity and completed versus committed %. Graphs were obtained from the quality manager in order to measure post sprint release defect arrival. The graphs date back to release

11, but for the scope of this thesis the researcher will only look at the graphs from release 15 onwards. The team implemented Scrum in release 13, but these graphs cannot be used as a comparison of the team before Scrum and after Scrum, as the way in which defects were logged was changed. Thus the data before release 15 is not a true reflection of defects.

For the customer satisfaction survey the researcher sent out a survey to all the teams in the company that implemented Scrum for a period of at least six months. According to Oats (2006), “The idea of a survey is that you will obtain the same kinds of data from a large group of people in a standardized and systematic way”.

The researcher worked with the IT manager and other Scrum Masters within the company to compile a questionnaire that provided the widest range of feedback possible. Oats (2006) states that a questionnaire is a set of pre-defined questions that was assembled in a predetermined order.

The questionnaire consisted of both open and closed questions, providing a source of both qualitative and quantitative data. The questionnaires were self-administered, thus the researcher was not present when the respondents completed the questionnaire.

The aim of the questionnaire was to get ground level feedback on all areas directly related to Scrum or that are impacted by Scrum. The team members could give feedback on their experience within the process, what they would like to improve, how it impacted them as a team and how it impacted them as individuals. The team members had the option to either fill in their names or to stay anonymous. All the other questions were compulsory. For the sample questionnaire, please refer to Appendix B – Scrum survey.

The questionnaire was compiled in an electronic format and a pilot was run with the Scrum Masters of the different teams. After incorporating the feedback into the survey, a link was e-mailed to all of the different domains within the company that implemented Scrum along with the domain owners. The link was emailed to 55 members, and the researcher received 24 responses. One of the responses was part of the pilot and contained bogus information, thus the researcher only included the results of 23 of the responses that were received.

5.1.2 Analysis of data

The survey responses were evaluated using a technique called key-words-in-context (KWIC) (Ryan et al., 2003). Researchers identify key words from the text, and then search the rest of the text for other occurrences of the text as well as the context that it was used in. Themes are identified by sorting the examples into piles with similar meaning (Ryan et al., 2003). The qualitative questions were used to compile graphs from and to give statistics about what percentage of the test population felt a certain way about things.

The researcher went through each of the answers and sorted the key words according to occurrence. As the questions were grouped, four main contexts emerged: process, implementation, team and individual. The researcher then sorted the different responses under these topics. In order to ensure content validity (i.e. that all the negative responses are not just from one respondent), the research initially used the last four digits of the response ID to identify who said what. The IDs were omitted in the final tables in order to ensure anonymity.

The researcher drew up graphs summarizing the different aspects of the sprints, for example a velocity graph, and then used the information that she had captured on the team in order to explain the fluctuations and anomalies within the data. The researcher also correlated the fluctuations on the graphs with changes as recorded by the case study.

The post release defect graphs obtained from the quality manager were also compared to events as recorded within the case study along with changes in velocity and sprints to obtain what caused the defects.

The researcher used the framework in order to calculate quantitative data on how well the team implemented Scrum. The researcher then explained these values and the reasons for her ratings.

5.1.3 Reliability and validity of the study

Kalermo et al. (2002) state that “Reliability in qualitative research refers to how well all the data has been taken into account, whether it is transcribed correctly, and how well the results reflect the respondents' thoughts”.

In order to ensure the validity of the study, the researcher had to ensure that she had collected enough information and that all the conclusions can be collaborated by the data. Since the researcher is part of the Scrum team that she evaluated, she had to be careful to keep the conclusions based on the data and not based on her own bias. The only way to ensure this was to provide a proper audit trail of all the collected information.

6. Data analysis

The researcher made use of a variety of metrics in order to measure the success or failure of the Scrum implementation within the data warehousing team as well as the changes caused by the implementation. In order to determine if the implementation was successful, the researcher developed a framework based on "Scrum smells". Scrum smells are elements of Scrum that are done wrong.

The researcher also identified four metrics with which she evaluated the team: quality, predictability, productivity and value. These metrics as well as the framework will be discussed in greater detail in the sections below.

6.1 A framework to evaluate the implementation of Scrum in a data warehousing team

The researcher evaluated the Scrum implementation according to four main categories namely artefacts, meetings, roles and the overall process.

6.1.1 Artefacts

This section will evaluate the artefacts and how well the team fared keeping them updated in line with Scrum. The researcher will also explain why she gave something a high or low rating. The artefacts were backlog management, sprint burndown and release backlog.

BACKLOG MANAGEMENT	**SCALE**
Is the product backlog shrinking instead of constantly growing?	MOSTLY
Is the backlog kept updated?	ALWAYS
Is the backlog estimated?	SOMETIMES
Are the stakeholder's requirements and desires captured in the backlog?	MOSTLY
Is the backlog a manageable size?	MOSTLY
Is the backlog visible?	ALWAYS

Table 12 Framework - backlog management

The backlog management formed part of the Product Owner's responsibilities. It was kept updated, but the stories were not always estimated and sometimes a story would go into sprint planning one with a zero value attached to it. backlog was captured in an online tool and all the team members had access to it. Initially the backlog wasn't broken down to the right levels. This links back to the team not having regular estimation meetings, but this will be discussed under the meetings section.

Out of a possible 18 points, the team managed 13, giving an average of 72.22% for backlog management.

SPRINT BURNDOWN	**SCALE**
Is the sprint burndown visible?	ALWAYS
Is the sprint burndown being updated by the team?	SOMETIMES
Does the team actually use the sprint burndown?	NEVER

Table 13 Framework - sprint burndown

The sprint burndown was drawn on the bottom corner of the task board as well as in electronic format on an excel spreadsheet. The team did not display any interest in the burndown. The Scrum Master was the only person updating it, except for the odd occasion when she left it, and a team member decided to update it. The team scored an average of 44.44 per cent for sprint burndown.

RELEASE BACKLOG	**SCALE**
Is the release backlog updated?	MOSTLY
Is the release backlog visible?	MOSTLY
Does the team/Product Owner use the release backlog as a prediction tool?	NEVER

Table 14 Framework - release backlog

The release backlog also formed part of the Product Owner's responsibilities. The team did not manage to reach any consistency (please see the section on predictability metrics for more information on this) and as such the Product Owner could not use it as a prediction tool. There was never any discussion with the team about their commitment for the next release. The team only found out what was coming up after the commitment had already been made, and as such they had no input into the process. As a result, the team only scored 44% for this section.

Out of a possible 36 points, the team only scored 21, giving them an average of 58.33% for managing the artefacts associated with the Scrum process. This implies that the team did not implement or make use of their artefacts correctly.

6.1.2 Meetings

As mentioned previously, there are a lot of meetings that form part of the Scrum process. This next section will evaluate how well the team did when it came to having the meetings and adhering to the protocol of things that should happen within those meetings. The meetings to be evaluated were Sprint planning 1, Sprint planning 2, the review, retrospective, daily Scrum and estimation meetings.

SPRINT PLANNING 1	**SCALE**
Are sprint planning 1 meetings always held?	MOSTLY
Is the team allowed to select the amount of work they want to take into the sprint?	SOMETIMES
Is a sprint goal defined?	SOMETIMES
Are the stories taken into the sprint estimated?	SOMETIMES
Does the team know enough about a story to take it into a sprint?	MOSTLY
Is there an explicit done defined for a story?	SOMETIMES
Does the team take availability into account when making a commitment?	NEVER

Table 15 Framework - sprint planning 1

Sprint planning meetings were held regularly most of the time, but moved around a lot, especially when it came to having planning meetings during the last week of a release. Team members would refuse to go to meetings because they believed that they had too much work.

There were occasions where a story was only estimated in sprint planning 1, or where the first task on a story was further analysis. These type of stories should not be taken into a sprint as the team does not know enough to make a commitment. The team did not have a 'done' defined until near the end of the Scrum process, and within the 'done' there were still some loopholes, as stories were moved to 'done' even though the team only later discovered that there were parts of the story missing.

A sprint goal was rarely defined, but on some occasions the Product Owner did emphasize what she wanted at the end of the sprint. The team never adjusted their commitment to cater for team availability.

The team only scored 8 out of 32 for conducting sprint planning 1, giving them an average of 38.1% for sprint planning 1.

SPRINT PLANNING 2	**SCALE**
Are sprint planning 2 meetings held regularly?	SOMETIMES
Are tasks decomposed so that they can be done in less than a day?	MOSTLY
Is the design sufficient so that the team knows what to do for the story?	MOSTLY
Does the team re-evaluate their commitment at the end of the planning session?	SOMETIMES

Table 16 Framework - sprint planning 2

Sprint planning 1 and 2 were usually held on the same day. Thus, if sprint planning 1 was not done, sprint planning 2 was not done either. The team went through a phase where they used their sprint planning 2 meeting as a proper design session. This changed into just using it as a task defining session in front of the task board, without doing any design.

The tasks were decomposed to a manageable level, except for stories that needed more analysis. The team would then make use of place holders until they could break the tasks down further. There were instances where a task was too big and it was not broken down further.

The team scored 6 out of 12 for this section, resulting in a 50% average.

REVIEW	SCALE
Does the review have a relevant and meaningful theme?	SOMETIMES
Do the relevant role players attend the review?	MOSTLY
Is the review at a relevant technical level for the attendees?	SOMETIMES

Table 17 Framework – review

The team struggled with the review because of the type of development that they do. The business value lies in the reports pulled from the tables that the team creates, but the team themselves does not create these reports. Thus in the review the team could only demo the tables that they had created. All the relevant role players attended the meeting, but there were disagreements on the technical level of the review, as some members wanted to show the tools while others felt that they should only show the tables. This was never resolved.

The team scored 4 out of 9 for this section, giving them an average of 44.44%.

RETROSPECTIVE	SCALE
Does the retrospective inspect the last sprint?	ALWAYS
Does the team derive value from the meeting?	SOMETIMES
Does an action that gets defined within the retrospective get done?	SOMETIMES
Are new impediments identified each sprint instead of recurring impediments?	SOMETIMES

Table 18 Framework - retrospective

The retrospective always evolved around the last sprint. A lot of times it was difficult to get people to commit to the retrospective and to actually derive value from it. A lot of times people did not get to say what they really wanted because of underlying personal issues within the team. Most of the impediments that were identified were recurring impediments instead of new impediments.

DAILY Scrum	**SCALE**
Is the daily ritual consistent?	MOSTLY
Does the meeting happen at the same time every day without skipping a day?	MOSTLY
Are there clear expectations for the Scrum meeting?	SOMETIMES
Do external stakeholders remain quiet at the Scrums?	ALWAYS
Are the daily Scrum meetings focused?	SOMETIMES
Does the team answer the three questions on what they did, what they were going to do and what was standing in their way?	SOMETIMES
Is the task board up to date?	SOMETIMES

Table 19 Framework - daily Scrum

The daily Scrum happened at 9:30 in the mornings except for the times when the Scrum Master wasn't there (in which case it did not happen due to the team dynamics). The daily Scrum turned into a reporting session for the Product Owner, and sometimes it was so chaotic that team members did not attend. The researcher did an experiment where for two weeks she did not go, and no one in the team noticed it or commented on it.

The team scored 52% for this section.

ESTIMATION MEETINGS	SCALE
Do the estimation meetings happen at regular intervals?	SOMETIMES

Table 20 Framework - estimation meeting

The estimation meetings rarely took place. They would be scheduled, but the team would be too busy to attend or the team would decide that it added no value to that particular sprint. The team scored 33% for this section.

A good indication of the low scores for the meeting section can be obtained by looking at the sprint lengths of the team as summarized by the following figure:

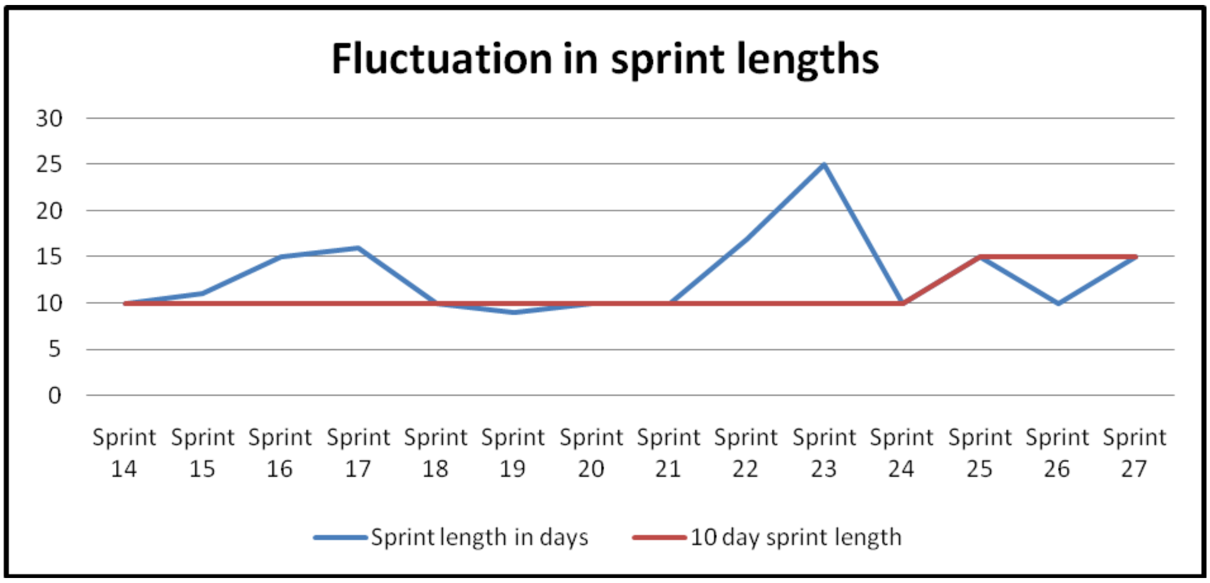


Figure 4 Fluctuations in sprint lengths

The 10-day sprint length is the baseline, and only on a few occasions were the sprints actually the same length as the baseline. The standard defined for the team was 10 days, and only from sprint 25 onwards did the team decide to move to a 3-week sprint cycle, which is why the baseline increased to 15.

6.1.3 Roles

As mentioned previously, there are different roles that will have to be fulfilled within the Scrum team. This section evaluated how well these roles were being fulfilled within the team. The roles that the researcher evaluated consisted of the team, the Product Owner and the Scrum Master.

TEAM	SCALE
Does the team understand what is expected from them during the daily Scrum meeting?	MOSTLY
If a team member cannot make a daily Scrum, do they inform someone in the team of work that was done?	SOMETIMES
Is the team committed to the Scrum process?	SOMETIMES
Does the team trust each other?	NEVER
Does the team honour team decisions?	SOMETIMES
Is the team accountable for the rest of the members in the team?	NEVER
Does the team do what it takes to reach their commitment?	MOSTLY
Does the team focus on team delivery above individual credit?	SOMETIMES
Does the team enforce completion criteria?	SOMETIMES
Does the team deliver consistently?	NEVER
Does the team take collective responsibility for not delivering?	SOMETIMES
Do team members volunteer for tasks?	MOSTLY
Is the team self-organizing?	MOSTLY
Does the team keep available team members in mind when making the commitment?	NEVER

Table 21 Framework - Team

The team had a clear understanding of what should happen in the morning Scrum since they had all been involved in the Scrum training. However, the communication within the team wasn't clear, and a task would remain on the board until the person responsible for the task came back.

In general there was a lack of trust within the team. As mentioned in the case study, management had to get a Scrum coach twice and the team admitted that they didn’t trust each other. There was a personality clash between role players as well as it being due to the shift in roles (analysts became developers, and developers did not trust the code from the analysts).

The team was self-organizing, arranging themselves according to personality. Management did not interfere in the organization of the team. Team members were allowed to pick their tasks, although sometimes they did not have a choice in the matter.

Team decisions were not always honoured. After a long meeting the team would decide on an action but when it came to doing it, people in the team would have changed their mind and not communicated it to the rest of the team.

The team did not deliver consistently. This will be discussed in greater detail in the predictability metrics section.

Product Owner	**SCALE**
Is the Product Owner available to the team during meetings?	MOSTLY
Does only the Product Owner maintain the product backlog?	ALWAYS
Is the Product Owner directly involved in feature definition?	MOSTLY
Is the release plan updated after sprint planning meetings?	ALWAYS
Does the Product Owner allow the team to commit without influencing the amount of work that the team commits to?	SOMETIMES

Table 22 Framework - Product Owner

The Product Owner was very adamant about what the team had to deliver in a sprint. It was very difficult to argue about being able to commit to less. The product backlog was the responsibility of the Product Owner alone and she kept it updated. The Product Owner was mostly involved in the feature definition and fed the information through to the team. The team scored 66.67% in this section.

Scrum Master	**SCALE**
Does the Scrum Master work diligently to enforce the rules?	ALWAYS
Has the Scrum Master set expectations for the daily Scrum?	MOSTLY
	SOMETIME
Is the Scrum Master in control of the meetings?	S
Has the Scrum Master been trained?	ALWAYS
Does the Scrum Master question dots?	MOSTLY
	SOMETIME
Is the Scrum Master's influence protected from outside forces?	S

Table 23 Framework – Scrum Master

Because of the dual roles, being Scrum Master and a junior team member in the team meant that the Scrum Master was not always in control of the meetings, since senior people overrode the junior people. This refers back to the self-organization in the team, as this typically should not happen. The team scored 66.67% for this section.

6.1.4 Overall process

This section refers to the Scrum process and how well the team followed it. The researcher looked at two aspects, namely how well the team did per sprint and impediments.

PER SPRINT	SCALE
Is all development that is taking place visible on the task board?	SOMETIMES
Do the external stake holders understand the process?	SOMETIMES
Are status reports only contained within sprint planning 1?	MOSTLY
Do the features and priorities of the sprint only change during sprint planning meetings?	SOMETIMES
Does the team constantly make use of the product backlog?	SOMETIMES
Does the team limit the amount of work in progress?	SOMETIMES
Are features only moved when fully done?	MOSTLY
Is a story done before the team starts working on other things?	MOSTLY
Is 'done' clearly defined for each feature?	MOSTLY
Is testing an integral part of feature definition?	SOMETIMES
Is progress visible?	ALWAYS
Is the feature that was selected independent of external dependencies?	MOSTLY
Are bugs fixed as they are found?	ALWAYS
Are sprints consistent, i.e. the same lengths?	ALWAYS
Does the task board reflect what the team is actually working on?	MOSTLY

Table 24 Framework - Sprint

The team went through a bad patch where much development was being done that was not visible on the task board. The reason for this was because team members were working on items that the team had not committed to, or people did not honour team decisions and redesigned packages instead of completing what they had committed to. The team scored 60% for this section.

IMPEDIMENTS	SCALE
Does the team properly identify impediments?	SOMETIMES
Does the team resolve impediments?	SOMETIMES
Does the team treat the actual root cause as opposed to symptoms of an impediment?	SOMETIMES

Table 25 Framework - Impediments

The team was too busy to actually focus on removing impediments. When the team did have a retrospective, they did identify impediments, but it was rare for these impediments to be taken into sprints as action items. The team scored 33% for this section.

6.1.5 Summary

Out of a total of 243 points, the team only scored 123 points, resulting in an average of 50.62%. As previously mentioned, in order to deem the implementation as successful, the team will have to score at least 80% on the framework. Thus the company did not implement Scrum successfully in the BI team.

Despite this, the researcher will evaluate the effect that Scrum had on the team in the next section, as there were still changes to process and the way in which the company does things.

6.2 Scrum metrics

As mentioned in Chapter 5, Beherns (2008) defines metrics across four dimensions when it comes to evaluating a team in Scrum: quality, predictability, productivity and value. The next section will evaluate the Scrum implementation based on these criteria.

6.2.1 Quality

Post Sprint defect arrival measures the number of defects that was found after the development team released code into the production environment. These defects are summarized in figure 5. As mentioned previously, the range that the researcher will be evaluating is from release 15 to release 18.

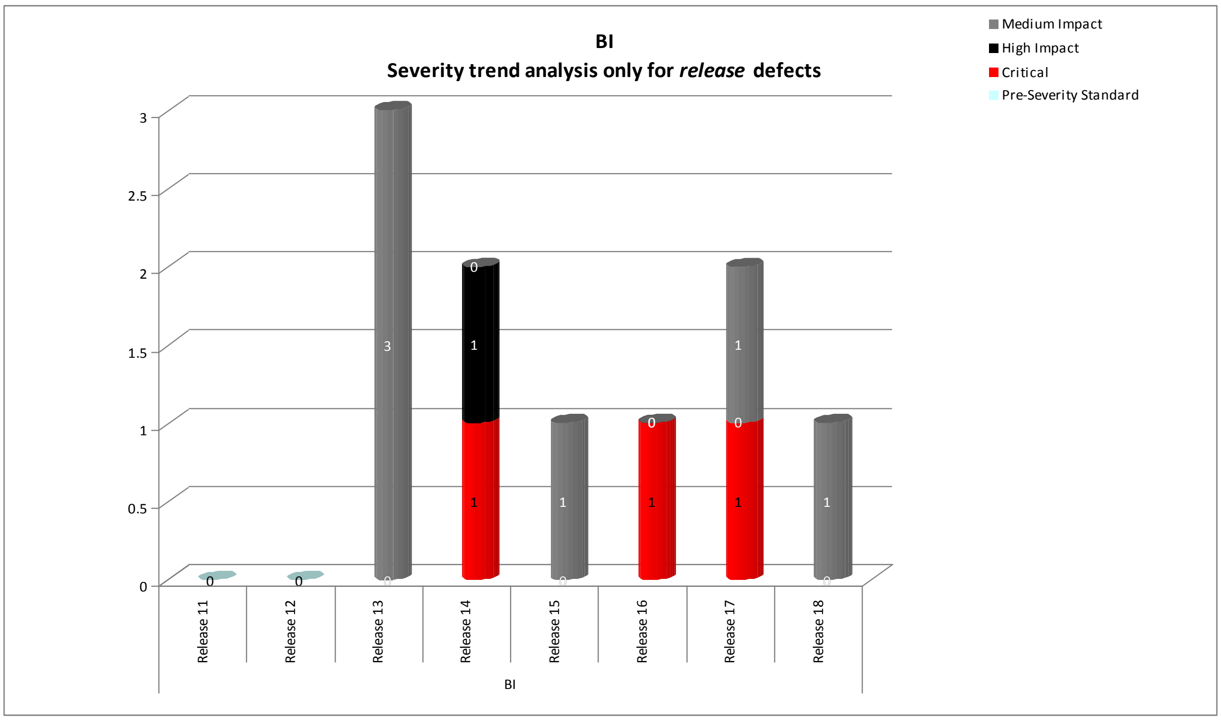


Figure 5 Post Release Defects

When logging defects, the team assigns different severities to each defect. A critical defect means that team has to stop everything that they are busy with and fix it immediately. A high impact defect will have to be fixed by the end of the day, and a medium impact defect can be fixed with scheduled production support, which usually goes out on Thursdays.

In release 15, the team only had one medium impact defect, in release 16 the team had one critical defect, and in release 17 the team had both a critical and a high impact defect. Release 17 was the release in which the team performed the worst, and it was also around the time when the team dynamic changed and analysts started to develop (Refer back to Chapter 4, Case study for more detail). In release 18 the team had improved their quality and only had one medium impact defect.

6.2.2 Predictability

The researcher used velocity to measure the predictability of the team. The velocity was measured from sprint 14 to sprint 27. After the first three sprints the researcher calculated the running average per sprint in order to determine how close the team was to committing to what they delivered in the previous three sprints. This is summarized in the graph below:

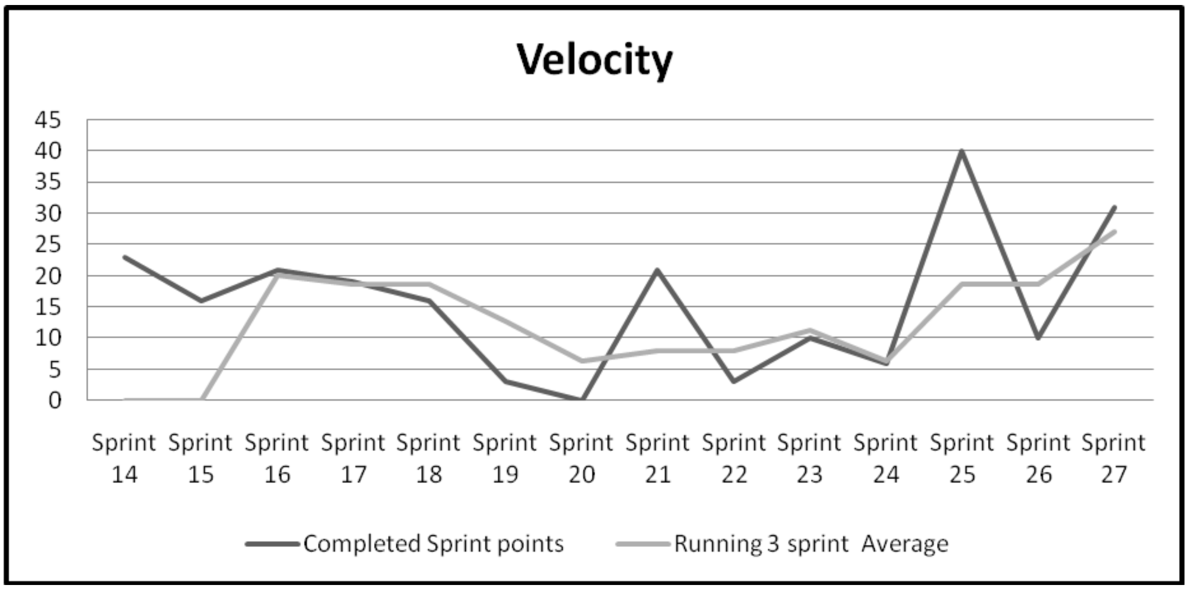


Figure 6 Velocity

The closer the two lines are together to each other, the more predictable the team has become. You would expect in time to see that the lines remain very close to each other, with only small fluctuations. As can be seen from the graph, the team continued to have major fluctuations in their velocity. The only times that the two lines came close to each other, were on the two occasions that the team was visited by the Scrum coach. This happened during sprint 16 - sprint 19, and from sprint 27 onwards.

On two occasions the team finished more work than they had committed to. The first time it happened was during sprint 21. In sprint 19 and sprint 20 the team finished little to no points and all these stories were completed with additional stories in sprint 21.

6.2.3 Productivity

The researcher decided to use the metric "committed versus delivered expressed as a percentage" as the metric to measure productivity. This metric is a bit controversial in the sense that if the team knows that they are being measured with this, they can easily manipulate their commitment so that they always play it safe (Beherns, 2008).

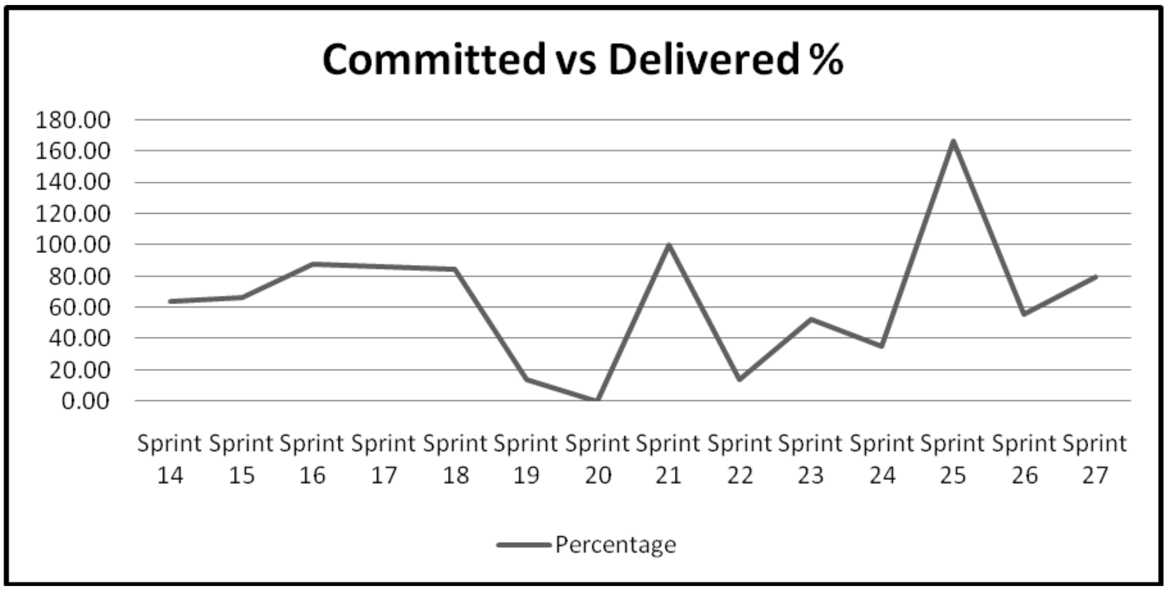


Figure 7 Committed versus Delivered

Ideally you would want the percentage as close to a 100% as possible. The team should constantly push themselves to improve, so there might be changes in what they commit to. In one sprint they might reach their commitment, the next sprint they commit to more, and miss it by a story. Thus you will have a fluctuation in the line, and never a pure 100%.

The problem with the BI team is that there is no constant delivery. The team constantly over-commits to the amount of work that they can do, and then does not deliver. The team crossed the 80% delivery line only 5 times in a total of 15 sprints, that is one-third of the time. And one of these spikes can be attributed to two sprints of not finishing work, and then finishing all the work at once.

6.2.4 Value

The last metric that the researcher used to evaluate the impact of Scrum with was an employee satisfaction survey. As mentioned previously, the questionnaire was designed to evaluate the Scrum process, implementation, teams and team members. The context was identified to be either a positive or a negative response. These areas will be discussed in more detail in the sections below.

6.2.4.1 Process

The respondents were asked to rate their experience of the Scrum process within the company on a scale of 1 to 5. A rating of 1 indicates a negative experience, a 3 is neutral, and 5 indicates that the user has had a very positive experience. The feedback is summarized in the following graph:

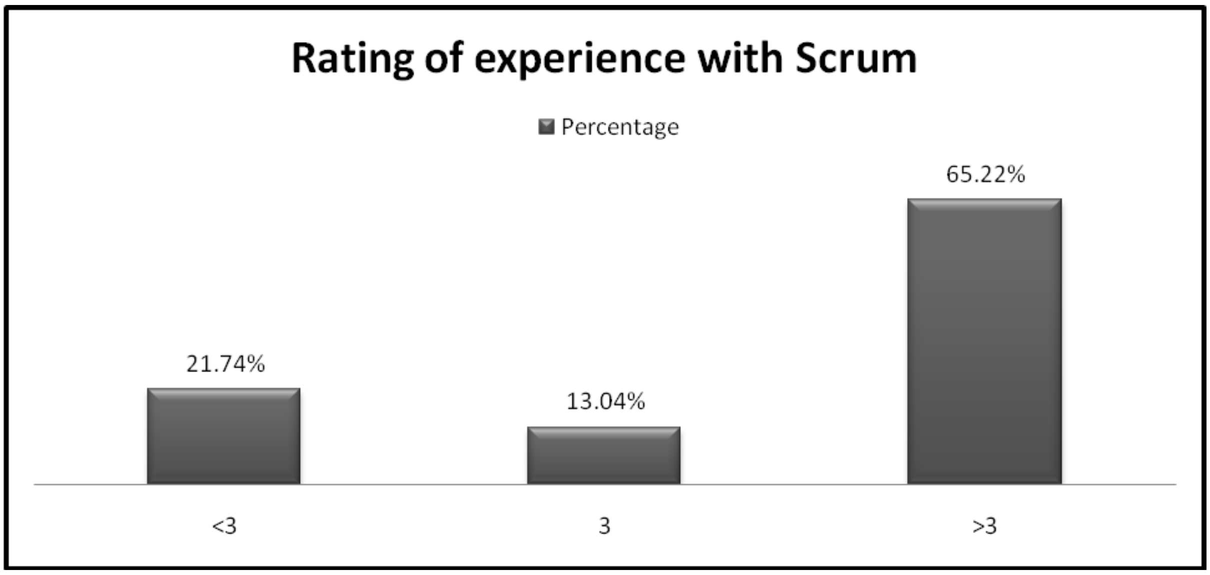


Figure 8 Rating of experience with Scrum

As can be seen from the graph, 65.22% of the respondents had a positive experience while 21.74% of the respondents did not have a good experience with Scrum. The remaining 13.04% of the respondents had a neutral experience.

The respondents were asked to give an explanation of their ratings as well as to list the positive experiences that they had with Scrum. The feedback and the emerging theme are summarized in table 25:

Emerging Theme	**Keyword**
Predictable	Predictability, Know what commit to, Can plan and schedule releases, Confidence in delivery dates, Better forecasting
Process Advantages	Stability, Focus, Structure, Provides a platform for knowledge sharing, Continuous improvement, Good at getting things done, Increased productivity, Emphasis on feedback and collaboration, Sprints work well
Visibility and Transparency	Visibility, Business knows where teams are, Know what other teams are doing, Transparency
Speed of delivery	Quick turnaround, Move from idea to delivery in a short time, More features quicker, Pace of change, Deliver value quickly, Continuous delivery, No long projects
Project output	Consistent flow of work, Manageable work, Impressive output, Less waste, Increased productivity
Team	Team has freedom to discuss issues retrospectively, Works as a team, Works very well for development teams, Visible goal, Provides direction for the team
Tangible	Tangible progress, Tangible delivery, Deliver value incrementally
Roles	Defined roles, Product Owner can focus on clients
Business	Relationship with business, Business can prioritize

Table 26 Explanation of ratings

Increased visibility and transparency were mentioned by most of the respondents as a positive experience that they had had with the Scrum process. They liked the fact that everybody was able to know where the team was and also what other teams were doing. They also commented on the predictability of the process and how easy it became when forecasting and planning releases.

The respondents were very happy with the output reached with the Scrum process. They also commented on the speed of delivery, the constant flow of delivered items and the fact that the delivery was tangible. Some of the respondents liked the team environment as well as the defined roles within the teams. Lastly, one respondent mentioned that the teams now had a better relationship with business. Eighty per cent of the respondents felt that the quality of the product that they delivered had increased from before. Seventy-two per cent of the respondents feel that they have been more productive since the company implemented Scrum.

The respondents were asked to explain their negative ratings, as well as to list any negative experiences that they had within the context of Scrum. The following key ideas were listed:

Emerging theme	**Keyword**
Scrum	• Doesn't work for some individuals, • Top down approach doesn't work, Top down approach ineffective, Unreasonable rules like number of stories in progress • Too many meetings, tedious meetings • Experienced developers are bypassed by juniors, Not buy-in from everyone • Constant pace of change overwhelming
Implementation	• People waste time arguing about what Scrum says, Scrum interpreted wrongly • No big picture view of the project, No plan • No documentation, Misunderstanding of requirements because of a lack of documentation • Not enough credit for non-feature work • Ineffective sprint planning 2 • Struggle to make commitments, Tied up by dependencies, Scrum hindered progress
Scaling Agile	Business expects a long-term commitment

Emerging theme	**Keyword**
Task Definition/ Implementation	Boring development tasks, Nothing you can sink your teeth into, Tasks are not challenging, Tasks are small, Takes joy out of development, Difficult when a task takes more time

Table 27 Negative experiences with Scrum

When evaluating the responses regarding the negative experiences that the respondents had with the Scrum process, the researcher had to distinguish between which of the responses were a result of Scrum itself, and which ones were a result of the way they were implemented. From the responses it was clear that the respondents attributed all the above-mentioned negatives to Scrum.

Some of the respondents felt that the top-down process that they have to follow when picking tasks and completing stories does not work for them. They also complained about the number of stories that are allowed to be in progress. They listed the many meetings as a negative, and one respondent commented that the constant pace of change gets to the team members. Another respondent mentioned that the process does not suit some of the people.

Some of the implementation negatives included that there is no big picture view, that the tasks are not well-defined and that the people do not enjoy completing the tasks, that the teams do not make their commitments and that there is a general lack of documentation or stability of requirements. This is a misconception that the company will have to address, because even though Agile is a methodology that believes in “just enough” documentation, it does not mean that documentation can be discarded. Some of the respondents mentioned that there seems to be a misunderstanding of Scrum within some of the teams. The implementation issues will be discussed in more details in the implementation section.

The respondents were asked if they would want to use any other methodology and to give reasons for it. The result is summarized in figure 9:

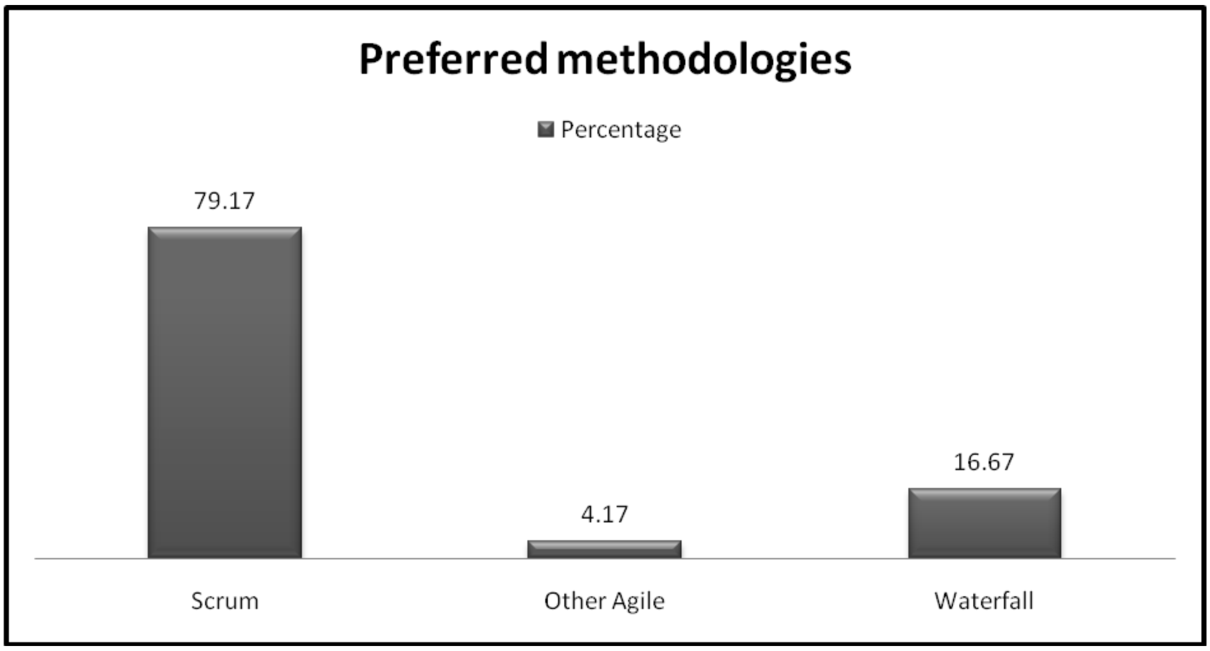


Figure 9 Preferred methodologies

Most of the respondents were happy with following some sort of Agile development methodology. Seventy-nine per cent of the respondents preferred Scrum and 4.17% would not mind following another Agile methodology, but with a focus on technology and not process. Only 16.67% of the respondents wanted to go back to the waterfall process.

The reasons that they listed for this is that proper documentation and reviews are available; that the developers stay ahead of the testers and that design issues are discovered before coding begins. One of the respondents said that you obtain greater job satisfaction and that it allows you to take ownership of work and also to take larger pieces of work. The reasons listed are implementation issues that the company will have to address.

6.2.4.2 Implementation

The respondents were asked if they felt that there were shortcomings in the way that the team implemented Scrum, and 72% of them felt that there were gaps in the way that the company implanted Scrum. The results were summarized as follows:

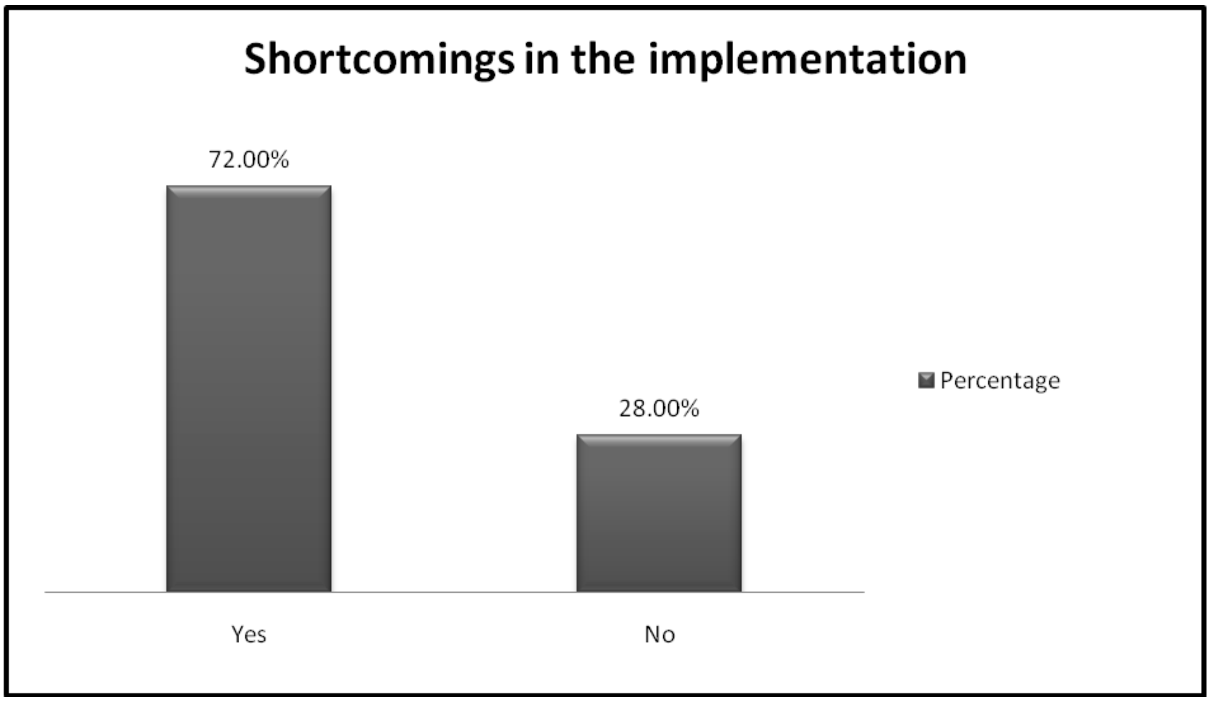


Figure 10 Shortcomings in the way that the company implemented Scrum

The respondents liked that the company seemed to be maturing within Scrum and that it is performing well. They identified and highlighted the following gaps with the implementation:

Emerging theme	**Keyword**
Dual roles	Conflict between being a Scrum Master and a team member, No dedicated Scrum Masters, Multiple roles
Align sprints	Need to align sprints between the different teams, sprints are not synchronized
Development process	Testing is rushed, not sure how testing fits in with Agile method, People procrastinate issues to later stories, Too much emphasis on making a sprint, Development gets rushed, Work too much in line, Specifications are not stable when developers get them, Extreme productivity has toll on people, Development work is tedious

Emerging theme	**Keyword**
Scaling Scrum	Problems with non-Scrum business structures, Lack of collaboration between IT and business, Not all teams use Scrum, Sprint reviews should be planned far in advance for business to attend, Training needs to be more involved, Rest of business does not understand Scrum, Slow buy-in from business, Business should be more involved
Coordination	Need to manage inter-domain dependencies, Dependencies on other teams, No coordination between teams, Integration issues, Teams are dependent upon other teams, Insufficient coordination between teams, Teams need more integration with other teams, Teams work in isolation, Overall delivery is somewhat painful, Developing silos
Basics	Process are not being driven, No commitment to implement actions, Team issues are not addressed, Don’t finish releases due to other impediments, Road to nowhere, No platform to address issues that go beyond the team, Not managing the Scrum of Scrums, Team struggled with implementation from start, Not all teams are applying Scrum 100%, Some teams perform well, others do not
Roles	Challenge to get team members to adopt other roles, Blurring of lines between developers and testers, Domain owners still involved in software delivery process, People are involved in areas they should not be
People	Not easy to give negative feedback to people you work with, People don’t buy into the process, People are scared to speak up

Table 28 Gaps in implementation

Most of the respondents commented on a lack of coordination and communication between the different Scrum teams. A result of this is the integration problems that the teams are experiencing as well as the painful deployment process.

The respondents commented on the lack of alignment between the start and end of the different Scrum teams. This made it difficult when teams are waiting on another team to finish work for them. They commented on the shared roles and expressed a need for dedicated Scrum Masters. They also highlighted the lack of dedicated Scrum Masters within some of the teams.

They commented that it doesn't feel like the process is being driven and that there is a lack of commitment to implement actions. Some of the teams are not applying Scrum correctly. There are also some team issues where it is difficult to address conflicts between team members.

Only 68 per cent of the respondents felt that there existed a platform where they could address both process and team issues. They named the retrospective as this platform. The remaining 32 per cent felt that these issues could not be addressed. The reason given for this was that it was hard to address team issues as you have to work closely with the team members and as such cannot address it without ending up in an uncomfortable situation. Others commented that sometimes issues are bigger than the team and that they don't have a platform where they can take those issues.

When asked what other issues they would like to be addressed, many of the respondents mentioned the lack of individual training and self-improvement, and that the company needed to provide them with the opportunity to enhance their skills. They also mentioned that there is a need to improve communication and coordination between the different teams.

Seventy-two per cent of the respondents felt that they did have the opportunity to implement the actions that were identified in the retrospective, while 28 per cent of the respondents felt that they did not have the chance to implement these actions.

The team members were also asked how they felt about multiplicity in teams. Multiplicity refers to one team member fulfilling two roles, i.e. a Scrum Master being a developer or business analyst as well. There was an almost level split, with 56 per cent of the respondents liking it, and 44 per cent of the respondents disliking it. The results are summarized in figure 11.

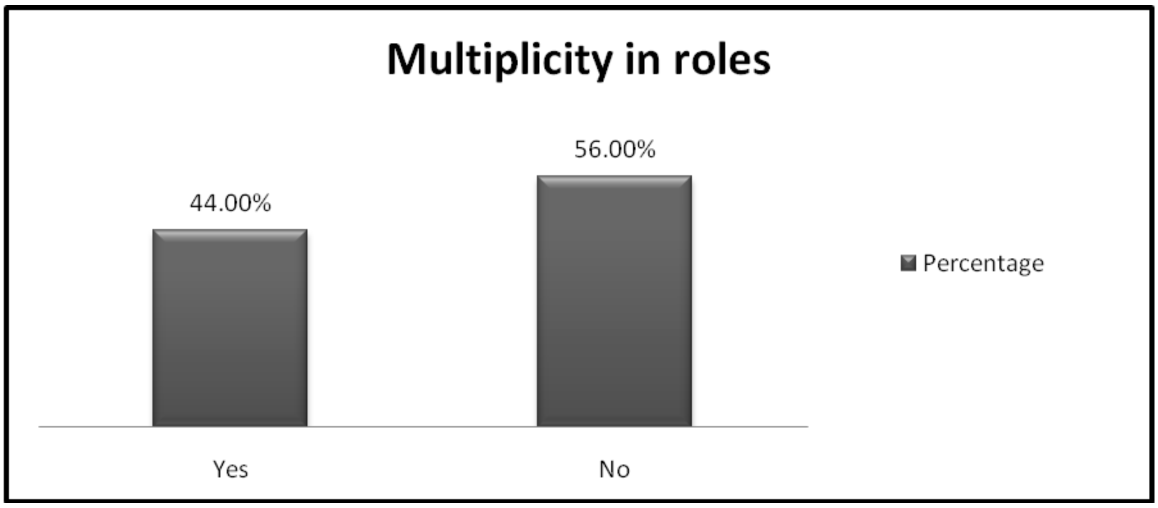


Figure 11 Multiplicity in roles

What was noticeable from the responses was that the people who were unhappy with multiplicity in the team were mostly developers. There seems to be some misconceptions about multiplicity in teams, and this seems to be a prerequisite. It is true that the lines between the different roles do become more blurred. Most of the people felt that they should stick to what they are trained to do, and that it is a waste of resources if you use your developers as testers. Others commented on the risk that it introduces when you have developers testing, especially when they test their own code.

Some team members mentioned that they had tried it in their teams, but that the quality of the testing deteriorated considerably. One of the respondents mentioned that multiplicity is unfair since it only affects certain roles within the teams, i.e. developers have to test but testers never become involved in development. This is something that the company will have to address.

6.2.4.3 *Team*

Some of the questions in the questionnaire took a look at how the team members experienced the team environment and mentality that are introduced into the software development process. The positives were:

Emerging theme	**Keyword**
Job satisfaction	People have adapted well, Most people are comfortable, Boosts moral, Team spirit
	Team interaction, People cooperate more, Team dynamic, Whole team is involved in solution, Team determines how, Team focuses on delivery process
Growth	Team learns, Team improves, Team knows what is going on, Knowledge is shared
Team work	Get a job done collectively, Team responsible for delivery, Collective effort, Satisfaction of completing features as a team, Teamwork

Table 29 Team positives

The respondents felt that most of the people adapted well and that there was a general boost in moral and team spirit. Many of the respondents enjoyed the team interaction and the fact that the whole team is involved in designing a solution. They also mentioned growth, i.e. the team learning, improving and sharing knowledge with each other. They also liked doing work as a team and taking collective responsibility for delivering a product.

When asked if they felt that the team was more productive, sixtyseven per cent of the respondents felt that there was an increase in team productivity since Scrum was implemented.

The negatives that the team experienced can be summarized as follows:

Emerging theme	**Key words**
Mind shift	Getting used to sharing responsibility, Need to learn to feel sense of achievement through output of team, Get used to being team-focused
Buy-in	Some people are not very involved in the process, People don't pitch up for meetings, People manipulate teams for own gain
Inward focus	Teams become inwardly focused
Dynamic	Struggle to get team dynamic right, Strong personalities in teams, Teams are not self-organizing

Table 30 Team negatives

Some of the respondents commented that it takes time to adjust to the Scrum process and the concepts of shared responsibility and feeling a sense of achievement through team output. It also requires a mind shift to get used to being team-focused. Three of the respondents commented that there is a struggle to get the right team dynamic, that some teams are not self organizing and that strong personalities in the teams have a negative impact on the team.

There is a feeling that not everybody is involved in the process and that people get to manipulate a team for their own gain. Lastly one of the respondents commented that the teams become too inwardly focused. This results in the communication and integration issues that were discussed above under implementation.

The respondents were asked how they felt about hierarchies within the team and 52.17 per cent felt that it wasn't a problem, while 47.83 per cent felt that it was a problem. The reasons given included that team members felt reluctant to disagree with reporting managers if they are in the team and that they are scared to raise issues as it might count against people in performance appraisals. This made it difficult for team members to discuss things in an open way.

6.2.4.4 Team members

The respondents listed the following positives about being in a Scrum team:

Theme	**Keyword**
Satisfaction	Job satisfaction, Feel like you achieve something, Allows you to grow
Pressure	Constant pressure
Improvement	Driven to improve, Team aims to continuously improve
Interaction	Improved people skills, Work closely with people
Learning	Knowledge gain, Learned a lot, Learn, Skills transfer

Table 31 Scrum team positives

They like the learning aspect of being in a Scrum team. There is a knowledge and skills transfer that happens implicitly. One of the respondents likes the constant pressure, and two of the respondents commented that they enjoy the drive to improve the team. Two of the respondents also liked the feeling of achievement and job satisfaction.

Some of the positives were counteracted by the negatives that the respondents mentioned. These responses are summarized as follows:

Theme	**Keyword**
Job satisfaction	No sense of achievement, Work not fulfilling, Not satisfying to do work, Feel like you don’t personally contribute, Doesn’t feel like I add much value, Feel like a cog in the wheel
Growth	Doesn’t give much room for trainees to grow
Loss of identity	Fear of loss of identity due to lack of titles

Table 32 Scrum team negatives

Four of the respondents felt that there was no job satisfaction anymore and that it didn’t feel like they added any value. One of the trainees mentioned that there was not a lot of space for the trainees to grow within the team. Some of the senior people feared a loss of identity or prestige due to the fact that there are no titles in the team.

The last thing the researcher evaluated was the retrospectives of the team and how successfully they improved their process and removed impediments. The next section will look at it in greater detail.

6.3 Evaluating the action research

The researcher captured all the actions as specified in the retrospectives and summarized these in the following tables. A tick mark indicates that an issue was resolved and a cross indicates that the team never solved it. A date in the done column indicates the next retrospective where this issue was raised again. The gaps in the dates can be explained by the team not having a retrospective or the team just having a discussion and not identifying any actionable issues.

6.3.1 19 March 2009

Issue	Done
Finish our historical regression tests (5 of 20+ tables currently tested)	✗
Schema change impact assessment	✗
Not enough data analysis (last week – found issues too late)	✗
Create capacity for future analysis	✗
Automation of daily processes (validation, error checking, broker patch, etc.)	✓
Assume less/ know more (assumed tests were easy to run and well-structured when they weren't)	✗
Plan as a team (last 2 days, better knowledge-sharing)	✓
Lack of commitment within the team (same hours when we were behind, urgency, be serious about our commitment)	✗
Better planning sessions with business (people with domain knowledge)	✗
Reaction time from IT ops is slow	✗
Keep tests up to date as we add new functionality	✓
Define IT ops role in all environments (e.g. UAT monitoring) to prevent delays	✗

Issue	Done
Standards (documentation to work from, coding, template, peer review)	✗
Code freeze e.g. adding more items too late Define policy	✗
Time to learn/grow (read articles, etc.)	✗
One deployment folder	22 June 2009

Table 33 Retrospective 19 March 2009

In the retrospective on 19 March the team identified gaps in their testing. It was decided that per sprint the team would take at least one table that was already created and write test cases for it. In the months that they were evaluated, the team did put a lot of effort into sorting out their testing. But until the end they were not satisfied with the work that they did, or with the test case coverage.

The one positive thing that the team did do is that for every new table or package that they defined, or for changes to an existing package, they included testing as part of what was done. Thus the test cases were updated continuously.

It took the team a long time to implement coding standards. They created a spreadsheet with the table definitions for each of the tables, and they also determined a standard for naming packages. However, the tasks in the packages did not always follow the standards, and this made reporting and logging of the packages quite difficult. The team did not implement a code freeze either, and often changes were made until two or three days before the release, or even on the day of the release.

The team continued to struggle against assumptions within the team. This can be attributed to reluctance within the team to do future analysis. The analysts were of the opinion that in doing the analysis you were doing the development, while the developers argued that you need the data analysis in order to identify anomalies within the data before you can do the actual development. This remained a hugely contested point until the end, with the team failing to reach agreement and this caused friction within the team.

The team continued to struggle in their interaction with IT operations. There was no clarity as to who is responsible for what and during what time frame. Even though each team knew what environments they were responsible for, environments like the user acceptance environment was a problem as it was not being monitored in the last two weeks of a release, and that was the time that this was critical to all the teams. This specific issue was solved by the SMS functionality that the team implemented on 13 October.

The lack of commitment also remained a contention point within the team. Some members felt that other members were slacking in their work hours and specifically when it was close to release time. If the team could not make a release, there was not a proper urgency or any extra work put in to make the release.

6.3.2 6 April 2009

Issue	**Done**
Should have two Scrums a day	✗
The team needs to go on a team build	✓
The team needs better hardware	✓
The network drive needs to get sorted out	✓
Code must be commented ○ Notes on why something was done ○ Need a documentation template	✗
Need to get a better integrated environment	✗

Table 34 Retrospective 6 April 2009

Despite making the decisions to have two Scrums during the day (one in the morning and one in the afternoons) the team only ever enforced this when it was the last week of the release and there was a lot of pressure on the team. The need for a team build was identified on 6 April 2009, but the team only ended up going on one in October 2009.

The team sorted out their folder structure and created one folder where they kept all their documentation organized per dimension. The team started to document what they had done and why they had done it, but this was never maintained and the team did not use the template that they ended up creating.

The team expressed the need for an integrated environment. All the database instances were reshuffled so that it all conformed to the same standard, but there was no full integration before the user acceptance testing level.

All the machines were upgraded to have 2 gigs memory, and each of the team members had two screens to facilitate the ease of testing and development.

6.3.3 25 May 2009

Issue	Done
The team needs to define a strategy for the release on what they want to deliver and where they are going with it	✗
The team needs to find key points for the smoke tests	✓
IT operations need to be on the mailing list for the preproduction environment	✓
There is a need for SMS notifications for when processes fail There is a need for a dashboard for the ETL that contains failures, spikes, space utilization and daily checks.	5 June 2009

Table 35 Retrospective 25 May 2009

In the beginning of a release, the Product Owner sits with the team and discusses what she expects from them for the release. The problem is that this is too late, as she has already made the commitment to business without consulting the team. The team expressed the need to be a part of the process, but they were never included in the decision.

The team identified key points that they wanted to test before deployment, and designed a package that they gave to IT operations. The team started to use this package as part of their deployment only on 13 October 2009.

The team identified a need for an IT operations person to be on the mailing list for the preproduction environment. This is the last environment before deploying the code into production, and it always contains the latest data. It is critical before a release that any code runs through in this environment. It is under the control of IT operations, thus the team needed someone from them to monitor the environment.

6.3.4 5 June 2009

Issue	Done
Pushing through environments	✓
ETL Run every night	✗
Need a better way to test	✗
Look at tools to automate deployments	✓
Only extract columns that will be used by the ETL	✓
Need SMS notification	✓

Table 36 Retrospective 5 June 2009

The team struggled with the way in which they pushed code through the different environments. The deployment process was initially a manual process consisting of a collection of scripts that needed to be run. The team automated this process and continued to improve on the process until the end. The latest product was an automated deployment package with SMS notifications in the event of failure or success, which did the archiving as well.

The team would like an environment where the ETL runs every night. But because of dependencies on the environment by other domains, the team could not achieve this. Any failure would mean that the other teams are unable to work.

The team did investigate tools to automate the deployments, but in the end they designed a custom package to do the work for them. They also made a rule that any new scripts to be added to the extract process would only contain the columns needed, and not the whole table from the source systems. As mentioned previously, the SMS notification was added to the deployment and later on to the ETL as well.

6.3.5 22 June 2009

Issue	Done
Build in a check to see if a LOAD ID changed	✗
Fix old history before checking the new process	✗
The team needs to build in validations that need to form part of the daily process. The current processes need to be evaluated and enhanced	✗
The team needs to define a design for error reporting and auditing, and the processes need to be changed	✗
The team wants reports on the performance of the ETL in term of data moved as well as time	✓
The team wants to streamline the ETL	✓
More intense testing covering a greater spectrum	✗
Source control	✗

Table 37 Retrospective 22 June 2009

The source control issue was raised by the team once again. The main concern regarding the source control software that the team is using is that it is not equipped to handle DTSX packages. These packages are generated from the graphical tool that the team uses to write their code in, and are xml-based representations of all the components that the team wrote.

It cannot merge the packages if two developers worked on the same package. There is now a lock system in place so that whenever you make a change to a package, you first have to get a lock, check the package out, make the change, and then submit it.

The problem is that this is a manual process. The developer needs to remember to update before placing any file within subversion. It adds extra complexity to the process, especially if changes were made to a locked package. The problem came in as well when the team deployed to other environments. Because of the lock, all packages are marked read only. The developer needs to manually go and uncheck the read only option on each of the packages. The team did experience quite a few deployment failures in the cases where they forgot to do this.

The second concern is that each of the packages is now marked as read-only by default. This means that the automatic archiving process (when the team deploys to another environment) cannot overwrite or archive the file. The team now has to manually mark packages so that it is not read only, adding more overhead.
The team did a lot of work on the ETL to streamline it and to improve the speed of the process. Validations were never built in to perform daily checks and the team did not check for load IDs either. This came back as an issue later on, as the team had production support issues because they did not have a check in place for load IDs.

6.3.6 13 October 2009

Issue	**Done**
Testing is still a big issue, as the automated testing solution is in place, but a lot of the test cases fail with known issues in the data	✗
Source control is an issue, especially if the team finishes stories so quickly	✗
They stability of the user acceptance testing environment is a problem. If the ETL fails, the team does not know about it, and thus the environment is unusable the next day	✓

Table 38 Retrospective 13 October 2009

Up till the end of the measuring period, the team still struggled to find a way to do their testing in a meaningful way. The team ended up with a lot of test cases that failed every time they ran it, but they were never given the time to clean them up. The team spent a lot of time going through test cases, but a lot of the test cases only ended up being marked as known issues, instead of their fixing the issues.

Source control remained an issue. It seemed that bugs still managed to slip in, despite there being source control. Settings on the packages were changed, without the person checking it in having made the change. The team realized that this happened because people made changes in the deployment folder (which was not tracked) and then merged those changes back to the source control system.

The team subsequently made the decision that all changes will only be made in the source control system and that only before a deployment the code will be moved to the deployment folder. Only the person that is responsible for deployment was allowed to alter anything in the deployment folder.

The team also decided that when a change was committed, the on-time number for the fix had to be placed in the comments section. On-time was the tool that the team used to keep track of defects. This will help when they work on concurrent stories. If the team fixed issues, each fix had to be committed separately.

In order to address the stability of the testing environment, the team implemented an SMS wrapper. This would notify them in the event of a failure and enable the team to fix the issue before the next day. The team also implemented a smoke test package that they can run before deploying to ensure the integrity of the database that they are deploying to.

7. Conclusion

Even though Scrum is working for most of the teams in the company, there is still a lot of work that needs to be done within the company. The relationship with business has improved a lot but they still need to be educated about the process, what they can expect and how Scrum will fit into their plans. Thus the Agile method needs to be scaled through the organization in order to achieve maximum benefit. There seems to be a need for dedicated Scrum Masters in order to solve the complexities of dual roles. This will also clear up issues mentioned around no one driving the process and teams implementing Scrum wrong.

Roles within the teams will have to be defined. Within Scrum the team members do work together a lot closer and they have to do whatever it takes to get a story done, but this does cause a blurring between the lines of the different roles. There seems to be a perception that developers have to test, even though there are many warnings about the risk it introduces, as well as comments about the unfairness of it, since multiplicity only seems to affect certain team members. The company needs to decide how to handle team issues within the team as well as what platforms to put in place if the teams cannot solve these issues.

Job satisfaction came up many times, and the company will have to find a way to keep their employees satisfied within the Scrum framework. Many respondents mentioned a lack of training. The company will also have to look at the way they define tasks and conduct the meetings in order to derive maximum value from it.

Based on what the researcher saw, she does believe that Scrum can be used to provide agility in data warehousing. It did not fail within the team because of a fault with the framework; it failed because of reluctance within the team to follow the process and also due to a lack of support from top management. Thus it failed because of people issues, not because of the area to which it was applied.

The team was able to define user stories and to implement them. The team did manage to eventually establish a way of estimating these stories. And the team did find a way to fit these stories into sprints. Thus, Scrum can be used to deliver agility in data warehousing.

It does seem as if Scrum can provide a balance between Agile and traditional development by not dictating the “what” and the “how” of the development process and only providing a basic framework to which the team needs to adhere. This leaves the team the flexibility of determining on their own how they will solve the problem.

It seems like Scrum offers a counter problem to traditional data warehousing. Where people often complain about too much documentation when it comes to traditional development, there seems to be a lack of documentation when it comes to Scrum. This can be due to the way in which the company chose to implement Scrum, but it does raise questions on how documentation should be handled. Scrum does not dictate documentation of the development methodology, thus leaving teams the flexibility to determine for themselves what and how they want to carry out documentation.

From the researcher’s point of view she experienced many frustrations in the implementation of Scrum within the organization, and specifically within her team. According to Schwaber et al. (2009) “Team members must have all of the skills necessary to create an increment of work”. The team did not lack the skills, but lacked the time and experience in testing, and did not have time to upskill due to the work load. This meant that when the team was close to a release, there was constant pressure on the team to finish their testing.

Despite having a solution in place, the team consistently mentioned testing as one of their impediments in the Scrum reviews, but it was never addressed. The team was promised a testing resource, but this never realised.

Also, since the two analysts turned into developers, there was a lack of analysis within the team. The developers never made the switch to become analysts as well. The one analyst also believed that there was no need for future analysis, while the researcher sees it as a problem area within the team.

Schwaber et al. (2009) also mentions that “There are no titles in Teams, and there are no exceptions to this rule”. Within the BI team, there was a high degree of seniority. The problem that the researcher faced was that she was a junior member. The opposition and conflict within the team came from a senior member who believed he had the seniority to override decisions made not only by the researcher, but also by the rest of the team. This impacted meetings as well as honouring team decisions. On several occasions that member would not pitch for a meeting or would make a change despite the team saying no to it.

Moe et al. (2010) stated that “Agile software development emphasizes that teams should be self-managed. However, Scrum and Agile methods offer no advice on how shared leadership should be implemented. If an organization believes in letting teams be more self-managing, great care must be taken in the implementation”.

Even though Scrum was implemented, titles were never addressed and in the researcher's opinion some of the senior people felt a fear for lack of recognition if there were no titles.

The team is supposed to be self-organizing (Schwaber et al., 2009).The team should not be told how they should turn the product backlog into workable software. However, the BI team was influenced by the Product Owner. When the team committed to work, it was already organized into a sprint according to priority set by the Product Owner.

The fact that the work was already organized into a sprint, created expectations from the Product Owner, and in the researcher's opinion it influenced the outcome of the commitment. The researcher did address this, but it did not change. The team constantly over-committed, never learning that they should commit to less and rather finish what they had committed to, than not reaching their commitment.

Schwaber et al. (2009) stated that one of the common mistakes that teams make when implementing Scrum is that they would rather change the process than change themselves. This mentality was deeply embedded within the team, and they would rather try to change the process, than the way that they work. This was particularly clear when the team had an argument about future analysis, and the analyst believed firmly that in doing analysis you do the work, so you don’t need to do it beforehand. As mentioned before, this remained a contention point within the team.

According to Schwaber et al. (2009) “A new Team often first realizes that it will either sink or swim as a Team, not individually, in this meeting. The Team realizes that it must rely on itself. As it realizes this, it starts to self-organize to take on the characteristics and behavior of a real Team.” The meeting mentioned in the above passage is the sprint planning 1 meeting. This never happened within the BI team. It was very much a drive for individual recognition, and the researcher wonders if it was because of the company values that included individual accountability. It is hard to work within a team where team members state that they don’t need the rest of the team. And if that team member is the key information resource on the project, then it is an issue. The BI team was very much a group of individuals, there was no team mentality within the team.

And that is the question that is bothering the researcher to this day: How “self-organizing” do you let a Scrum team be? All the indications were there that the team was struggling. Yes, they delivered, but they did not deliver constantly, they failed to establish any velocity in over a year and they had to reset their estimations twice.

Shouldn’t management then step in, especially if it is clear that the team is not organizing itself correctly? The same issues were raised again and again, and yet no one did anything about them.

The second thing that bothers the researcher is this: “How do you handle strong personalities within a team?” The one thing that the researcher experienced was that team decisions were very much influenced by strong personalities within the team. The same issue would be raised multiple times, and yet, because one person in the four person team feels strongly about it, it is not addressed.

The researcher experienced first-hand the withdrawal from the sessions. The members became tired of being ignored or overridden, and in the end they just stopped arguing. That means that an issue did not get resolved despite attempts to do so. This is the one issue that is still of great concern to the researcher, and to which she has not been able to find a satisfying answer.

In the problem statement the researcher stated that she had hoped that the Scrum process would create more generalists. The reality is though that it creates great risk for the business. How do you handle the learning curve in people becoming generalists? The researcher was caught in a situation where an analyst decided to become a generalist. This meant that he did analysis, development and testing. Any development team can elaborate about the risks of this approach, as things are missed and are not analysed.

The bigger problem was that the team all of a sudden lost a key resource. The analyst was involved in all the meetings up-front, not the team. So one member had all the information, and in that got the power to not only dictate to the team, but also to only feed them information that he wanted them to have.

This tie in with people wanting to be a specialist in their field. Do you not put them into a Scrum team? Is it wrong for a developer to want to develop and not test? A change should be driven throughout the team and should be to the benefit of a team, not an individual. Sadly, there was too much individualism going on within the BI team.

In the survey on the state of Agile method (VersionOne, 2009), over ten per cent of the respondents contributed to the failure of Agile projects to lack of management intervention and an unwillingness of the team to follow Agile practices. The researcher believes that these were the root causes that caused the team to fail at implementing Scrum.

Despite the failure, the team was still able to experience some of the benefits of Scrum. Most of the benefits mentioned in the literature review were mentioned by the team members. It will be interesting to see how the team is doing a year from now, if they did manage to adopt the process, or if they opted for another methodology.

In order for Scrum to work in a team, the researcher would suggest the following:

1. Get buy-in from top management. Without buy-in from the managers, the implementation of Scrum will fail.

2. Monitor the team. This can be best achieved by having an external Scrum Master who is not influenced by team dynamics. Have metrics in place, allow the team to self organize, but make sure that the team organizes itself in a healthy and viable way.
3. Make sure that the team is cross-functional. By cross-functional, the researcher means that all the skill sets needed are present within the team. In the case of the team that was evaluated, four people were responsible for analysis, development, testing as well as the role as Scrum Master.
4. Team dynamics is very important. Because of the close interaction within the team, it is important that the team members get along well.
5. Have someone to push and drive the process. It is an active process that needs to be looked after and constantly improved. This is done by the Scrum Master, but he/she should be given the authority to be able to do this. Also, management needs to be involved and should also push for it.
6. Continuous improvement. Give the teams the time and capacity to improve, and reap the rewards.

In terms of approach, the researcher does believe that action research is the right approach to follow when evaluating a team in data warehousing. Scrum is about inspecting and adapting (Schwaber et al., 2009), so it makes it easy to implement action research.

It is important that the researcher is given the authority to be able to push and drive the actions specified. Even though the researcher was a part of the team, she often lacked the authority to be able to push for a change.

This is a new and exciting field, and there are many opportunities for further research. The next section details areas of research that the researcher would like to see, and hopefully someday get the opportunity to participate in.

7.1 Further research

This is a new and exciting field, and there are many opportunities for further research. As mentioned above, it will be interesting to evaluate the team after a year again to see what has changed. It will also be interesting to see it implemented in another team and to see if these teams experience the same issues and problems.

There were a lot of variables in the case study, and it will be really difficult to generalize. But by evaluating a few teams, the researcher believes that you will be able to determine what factors play a role in the success or failure of Agile data warehousing, and also if these factors correlate to the problems that the normal development teams face or if they are unique to a data warehousing environment.

There is also work to be done in determining metrics to determine the effect of Scrum, as well as measuring the implementation, as it is currently a very subjective approach. The problem with statistics is that if teams know how they will be measured, they will know how to change the system. Something will have to be in place that is fair and non-threatening to teams.

It will also be interesting to look at the set-up of Scrum teams in terms of personalities. According to Schwaber et al. (2009) “However, the skills that Team members share – that is, the skill of addressing a requirement and turning it into a usable product – tend to be more important than the ones that they do not. People who refuse to code because they are architects or designers are not good fits for Teams. Everyone chips in, even if that requires learning new skills or remembering old ones.” This could indicate the possible use of personality tests in the composition of teams or other techniques to ensure the cohesion of the team.

But the reality is that there are specialists in a team and people chose a field for a reason (i.e. a developer wants to be a developer, not a tester). So you have that natural resentment to shifting roles. It also seems that some people are more impacted, i.e. developers test but testers don’t develop, making it unfair. It will be interesting to find a way to address the construction of a Scrum team.

8. Bibliography

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002, "Agile Software development methods", *VTT publications,* pp. 1-112.

Baskerville, R. & Wood-Harper, A.T. 1996 "A Critical Perspective on Action Research as a Method for Information Systems Research," *Journal of Information Technology*, (11) 3, pp. 235-246.

Baskerville, R. & Wood-Harper, A.T. 1998, "Diversity in information systems action research methods", *European Journal of Information Systems,* vol. 7, pp. 90-107

Baskerville, R. & Pries-Jeje, J. 1999, "Grounded action research: a method for understanding IT in practice", *Accounting Management and Information technologies,* vol. 9, pp. 1-23.

Baskerville, R. & Myers, M.D. 2004, "SPECIAL ISSUE ON ACTION RESEARCH IN INFORMATION SYSTEMS: MAKING IS RESEARCH RELEVANT TO PRACTICE", *MIS Quarterly,* vol. 28, no. 3, pp. 329-335.

Beherns, P. 2008, "Scrum Metrics and Myths", [Online]. Available from: *Scrumorlando09.pbworks.com/Scrum-Metrics-and-Myths*. [2010/05/15].

Beyer, M.A. & Richardson, J. 2010, "Agile Techniques Augment But Do Not Replace Business Intelligence and Data Warehouse Best Practices", *Gartner*, pp. 1-8.

Beynon-Davies, P. & Williams, M.D. 2003, "The diffusion of information systems development methods", *Journal of Strategic Information Systems,* vol. 12, pp. 29-46.

Brobst, S., McIntire, M. & Rado, E. 2008, "Agile Data warehousing with integrated sandboxing", *Business Intelligence Journal,* vol. 13, no. 1, pp. 14-22.

Clutterbuck, P., Rowlands, T. & Seamons, O. 2009, "A Case Study of SME Web Application Development Effectiveness via Agile Methods", *Electronic Journal Information Systems Evaluation,* vol. 12, no. 1, pp. 13-26.

Dingsoyr, T., Dyba, T. & Moe, N.B. 2010, *Agile Software Development: Current Research and Future Directions,* 1st edition, Springer, Berlin Heidelberg.

Dyba, T. & Dingsoyr, T. 2008, "Empirical studies of Agile software development: A systematic review", *Information and Software Technology,* vol. 50, pp. 833-859.

French, S. 2009, "Action research for practising managers", *Journal of Management Development,* vol. 28, no. 3, pp. 187-204.

Gardner, S.R. 1998, "Building the data warehouse", *COMMUNICATIONS OF THE ACM,* vol. 41, no. 9, pp. 52-60.

Gibbs, W.I. 1996, "2.4 The Linear Sequential Model" in *Software Engineering: A Practitioner's Approach* Pressman & associates inc.

Guo, Y., Tang, S., Tong, Y. & Yang, D. 2006, "Triple-Driven data modeling methodology in data warehousing: a case study", *DOLAP'06*, pp. 59.

Hughes, R. 2008, " Agile Data Warehousing: Delivering World-Class Business Intelligence Systems Using Scrum and XP".

Introna, D. 1996, "Notes on ateleological information systems development", *Information Technology & People,* vol. 9, no. 4, pp. 20-39.

Introna, D. & Whitley, E.A. 1997, "Triple-Driven Data Modeling Methodology in Data Warehousing: A Case Study", *Logistics information management,* vol. 10, no. 5, pp. 235-245.

Ionel, N. "CRITICAL ANALYSYS OF THE Scrum PROJECT MANAGEMENT METHODOLOGY", *The Academy of Economic Studies Bucharest,* [Online], .

Available from: http://steconomice.uoradea.ro/anale/volume/2008/...management.../077.pdf. [2009/01/01].

Iversen, J.H., Mathiassen, L. & Nielsen, P.A. 2004, "MANAGING RISK IN SOFTWARE PROCESS IMPROVEMENT: AN ACTION RESEARCH APPROACH1", *MIS Quarterly.,* vol. 28, no. 3, pp. 395.

Kalermo, J. & Rissanen, J. 2002, *Agile software development in theory and practise*, University of Jyvaskyla.

Kimball, R. 2009, "Design Tip #111 Is Agile Enterprise Data Warehousing an Oxymoron? ", [Online]. Available: www.rkimball.com/html/09dt/DT111AgileEnterpriseDWOxymoron.pdf [2009, 05/05]

Lindgren, R., Henfridsson, O. & Schultze, U. 2004, "DESIGN PRINCIPLES FOR COMPETENCE MANAGEMENT SYSTEMS: A SYNTHESIS OF AN ACTION RESEARCH STUDY1", *MIS Quarterly,* vol. 28, no. 3, pp. 435-473.

List, B., Bruckner, R.M., Machaczek, K. & Schiefer, J. 2002, "A Comparison of Data Warehouse Development Methodologies: Case Study of the Process Warehouse", *Lecture notes in Computer Science,* vol. 2453, pp. 203-215.

Mahnic, V. & Drnovscek, S. 2005, "Agile Software Project Management with Scrum", *EUNIS 2005 Conference,*University of Manchester.

Martensson, P. & Lee, A.S. 2004, "DIALOGICAL ACTION RESEARCH AT OMEGA CORPORATION", *MIS Quarterly,* vol. 28, no. 3, pp. 507-537.

Meso, P., Madey, G., Troutt, M.D. & Liegle, J. 2006, "The knowledge management efficacy of matching information systems development methodologies with application characteristics—an experimental study", *The Journal of Systems and Software,* vol. 79, pp. 15-28.

Moe, N.B ., Dyba, T. & Dingsoyr, T. 2010, "A teamwork model for understanding an Agile team: A case study of a Scrum project", *Information and Software technology,* vol. 52, pp. 480-491.

Moss, L.T. 2001, "Business Intelligence Methodologies: Agile with Rigor?", *The journal of Information Technology Management,* vol. 14, no. 12, pp. 19-26.

Moss, L.T., Kelly, C., Rehm, C., Howard, S. & Tannenbaum, A. 2010, *What is the difference between the terms "business intelligence" and "data warehousing?"*, Information Management Online edition, Information Management and SourceMedia. Available from: http://www.information-management.com/news/7260-1.html

Nandhakumar, J. & Avison, J.E. 1999, "The fiction of methodological development: a field study of information systems development", *Information Technology & People,* vol. 12, no. 2, pp. 176-191.

Oates, B.J. 2006, "*Researching information systems and computing",* First edn, Sage Publications, London.

Orlikowski, W.J. & Baroudi, J.J. 1991, "Studying Information Technology in Organizations: Research Approaches and Assumptions", *Information Systems Research,* vol. 2, no. 1, pp. 1-27.

Rising, L. & Janoff, N.S. 2000, "The Scrum Software Development Process for Small Teams", *IEEE SOFTWARE.*

Ryan, G.W. & Bernard, H.R. 2003, "Techniques to Identify Themes in Qualitative Data", *Field Methods,* vol. 15, no. 1, pp. 85-109.

Schwaber, K. 1995, "Scrum Development Process", *OOPSLA'95 Workshop on Business Object Design and Implementation*Ken Schwaber.

Schwaber, K. & Sutherland, J. 2009, *Scrum guide* [Homepage of Scrum.Org], [Online]. Available: http://www.Scrum.org/Scrumguides [2009, 10/10] .

Sumner, M.R. 1985, "How should applications be developed", *DATA BASE,* , pp. 25-34.

Sutherland, J. 1999, *Nokia Test*, CSM v9.6 edn, Jeff Sutherland. [Online] Available from: http://jeffsutherland.com/ScrumButtTest.pdf. [2009/05/05]

Sutherland, J. 2001, "Agile *Can* Scale: Inventing and Reinventing Scrum in Five Companies", *The journal of Information Technology Management,* vol. 14, no. 12, pp. 5-11.

Sutherland, J. 2004, "AGILE DEVELOPMENT: LESSONS LEARNED FROM THE FIRST Scrum", *Cutter Agile Project Management Advisory Service: Executive Update,* vol. 5, pp. 1-4.

Truex, D., Baskerville, R. & Travis, J. 2000, "Amethodical systems development: the deferred meaning of systems development methods", *Accounting Management and Information technologies,* vol. 10, pp. 53-79.

VersionOne "2009 State of Agile Development", [Online]. Available from: http://pm.versionone.com/StateOfAgileSurvey.html. [2010/09/01].

Wake, W.C. 2005,*Scrum Process Mechanics* [Homepage of William C. Wake], [Online]. Available: www.xp123.com [2010, 05/05] .

Wixom, B.H. & Watson, H.J. 2001, "An Empirical Investigation of the Factors Affecting Data Warehousing Success", *MIS Quarterly,* vol. 25, no. 1, pp. 17-41.

Appendix A – Scrum Evaluation Framework

ARTEFACTS

BACKLOG MANAGEMENT	SCALE
Does the product backlog shrink instead of constantly growing?	NEVER
Is the backlog kept updated?	NEVER
Is the backlog estimated?	NEVER
Are the Stakeholder's requirements and desires captured in the backlog?	NEVER
Is the backlog a manageable size?	NEVER
Is the backlog visible?	NEVER

SPRINT BURNDOWN	SCALE
Is the Sprint Burndown visible?	NEVER
Is the Sprint Burndown being updated by the team?	NEVER
Does the Team actually use the Sprint Burndown?	NEVER

RELEASE BACKLOG	SCALE
Is the release backlog updated?	NEVER
Is the release backlog visible?	NEVER
Does the Team/Product Owner use the release Backlog as a prediction tool?	NEVER

Meetings

SPRINT PLANNING 1	SCALE
Are Sprint Planning 1 meetings always held?	NEVER
Is the team allowed to select the amount of work they want to take into the sprint?	NEVER
Is a Sprint goal defined?	NEVER
Are the stories taken into the sprint estimated?	NEVER
Does the team know enough about a story to take it into a Sprint?	NEVER
Is there an explicit done defined for a story?	NEVER
Does the Team take availability into account when making a commitment?	NEVER

SPRINT PLANNING 2	SCALE
Are Sprint Planning 2 meetings always held?	NEVER
Are tasks decomposed so that they can be done in less than a day?	NEVER
Does enough design happen so that the team know what to do for the story?	NEVER
Does the Team re-evaluate their commitment at the end of the planning session?	NEVER

REVIEW	SCALE
Does the Review have a relevant and meaningful theme?	NEVER
Do the relevant role players attend the review?	NEVER
Is the Review at a relevant technical level for the attendees?	NEVER

RETROSPECTIVE	SCALE
Does the retrospective inspect the last Sprint?	NEVER
Does the team derive value from the meeting?	NEVER
Does an action that gets defined within the retrospective get done?	NEVER
Is new impediments identified each sprint instead of recurring impediments?	NEVER

DAILY Scrum	SCALE
Is the daily ritual consistent?	NEVER
Does the meeting happen at the same time every day without skipping a day?	NEVER
Are there clear expectations for the Scrum Meeting?	NEVER
Do external stakeholders remain quiet at the Scrums?	NEVER
Is the Daily Scrum meetings focussed?	NEVER
Does the team answer the 3 questions of what they did, what they are going to do and what is standing in their way?	NEVER
Is the task board up to date?	NEVER

ESTIMATION MEETINGS	SCALE
Does the estimation meetings happen at regular intervals	NEVER

ROLES

TEAM	SCALE
Does the team understand what is expected from them during the Daily Scrum meeting?	NEVER
If a team member cannot make a Daily Scrum, do they inform someone in the team of work that was done?	NEVER
Is the team committed to the Scrum Process?	NEVER
Does the team trust each other?	NEVER
Does the team honour team decisions?	NEVER
Is the team accountable for the rest of the team?	NEVER
Does the team do what it takes to reach their commitment?	NEVER
Does the team focus on team delivery above individual credit?	NEVER
Does the team enforce completion criteria?	NEVER
Does the team deliver consistently?	NEVER
Does the team take collective responsibility for not delivering?	NEVER
Do Team members volunteer for tasks?	NEVER
Is the Team self-organizing?	NEVER
Does the team keep Team member available in mind when making the commitment	NEVER

PRODUCT OWNER	SCALE
Is the Product Owner available to the team during meetings?	NEVER
Does only the Product Owner maintain the product backlog?	NEVER
Is the Product Owner directly involved in feature definition?	NEVER
Is the release plan updated after Sprint planning meetings	NEVER
Does the product owner allow the Team to commit without influencing the amount of work that the team commits too?	NEVER

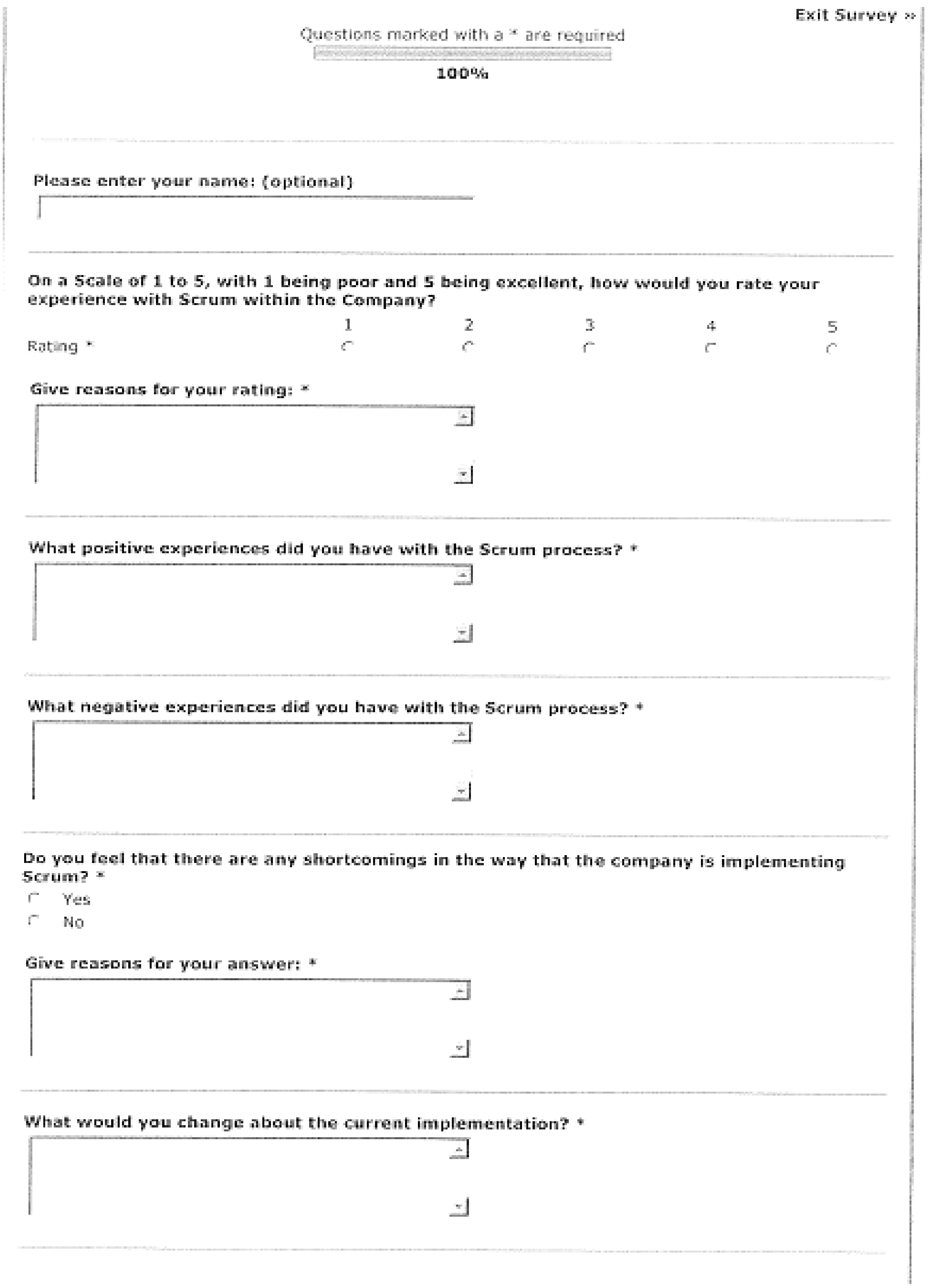
ScrumMASTER	SCALE
Does the ScrumMaster work diligently to enforce the rules?	NEVER
Has the ScrumMaster set expectations for the Daily Scrum?	NEVER
Does the ScrumMaster have control of the meetings?	NEVER
Has the ScrumMaster been trained?	NEVER
Does the ScrumMaster question dots?	NEVER
Is the ScrumMaster’s influence protected from outside forces?	NEVER

OVERALL PROCESS

PER SPRINT	SCALE
Is all development that is happening visible on the task board?	NEVER
Does the external Stake Holders understand the process?	NEVER
Are status reports only contained within Sprint Planning 1	NEVER
Do the features and priorities of the Sprint only change inside Sprint Planning meeting?	NEVER
Does the team constantly make use of the product backlog?	NEVER
Does the team limit the amount of work in progress?	NEVER
Are features only moved when fully done?	NEVER
Is a story done before the team start working on other things?	NEVER
Is Done clearly defined for each feature?	NEVER
Is testing an integral part of feature definition?	NEVER
Is progress visible?	NEVER
Is the feature that was selected independent of external dependencies?	NEVER
Are bugs fixed as they are found?	NEVER
Are sprints consistent, i.e. the same lengths?	NEVER
Does the task board reflect what the team is actually working on?	NEVER

IMPEDIMENTS	SCALE
Does the team properly identify impediments?	NEVER
Does the team resolve impediments	NEVER
Does the team treat the actual root cause as opposed to symptoms of an impediment?	NEVER

Appendix B-Scrum Questionnaire



Exit Survey »

Questions marked with a * are required

100%

Please enter your name: (optional)

On a Scale of 1 to 5, with 1 being poor and 5 being excellent, how would you rate your experience with Scrum within the Company?

	1	2	3	4	5
Rating *	○	○	○	○	○

Give reasons for your rating: *

What positive experiences did you have with the Scrum process? *

What negative experiences did you have with the Scrum process? *

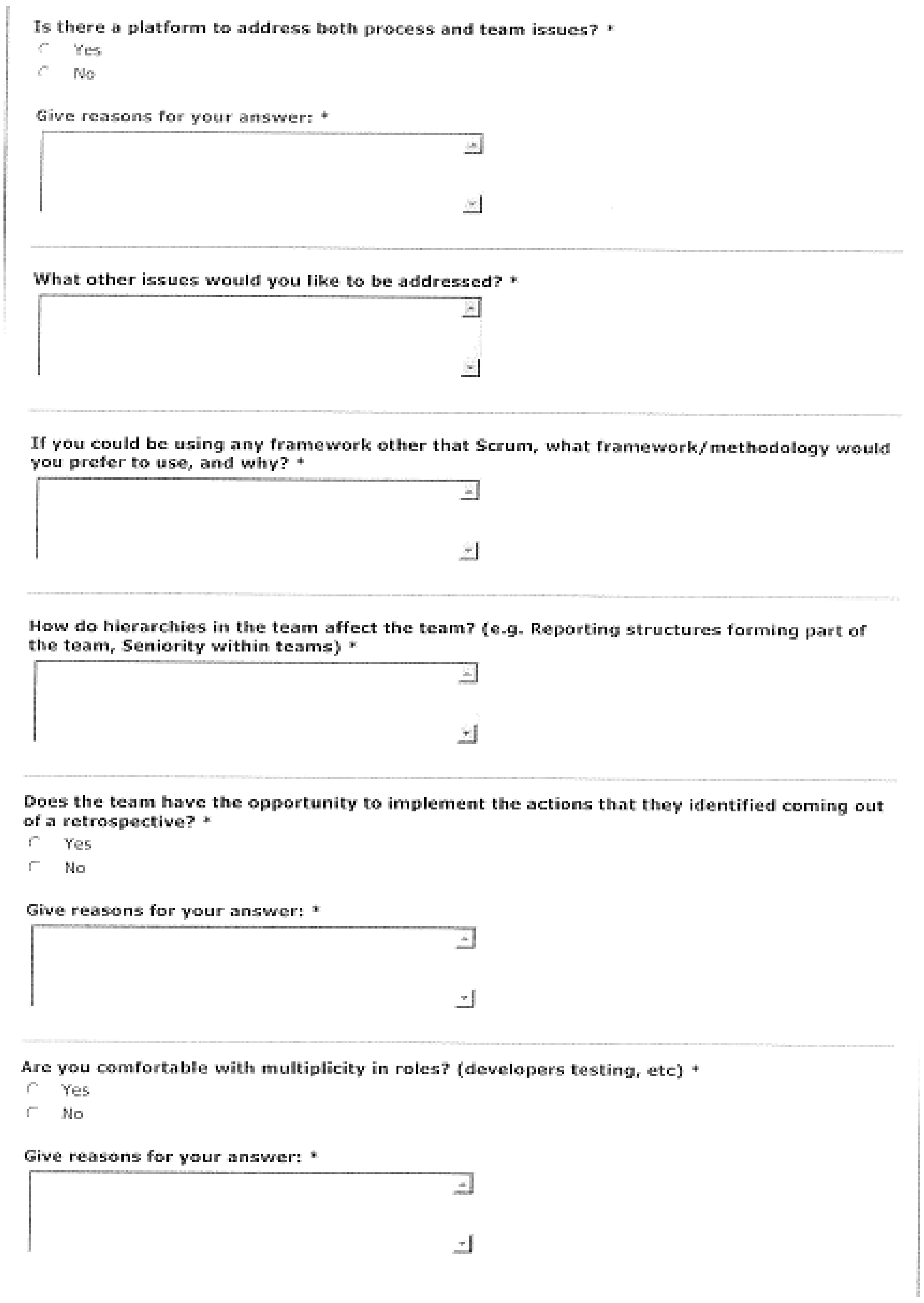
Do you feel that there are any shortcomings in the way that the company is implementing Scrum? *

○ Yes

○ No

Give reasons for your answer: *

What would you change about the current implementation? *



Is there a platform to address both process and team issues? *

- Yes
- No

Give reasons for your answer: *

What other issues would you like to be addressed? *

If you could be using any framework other that Scrum, what framework/methodology would you prefer to use, and why? *

How do hierarchies in the team affect the team? (e.g. Reporting structures forming part of the team, Seniority within teams) *

Does the team have the opportunity to implement the actions that they identified coming out of a retrospective? *

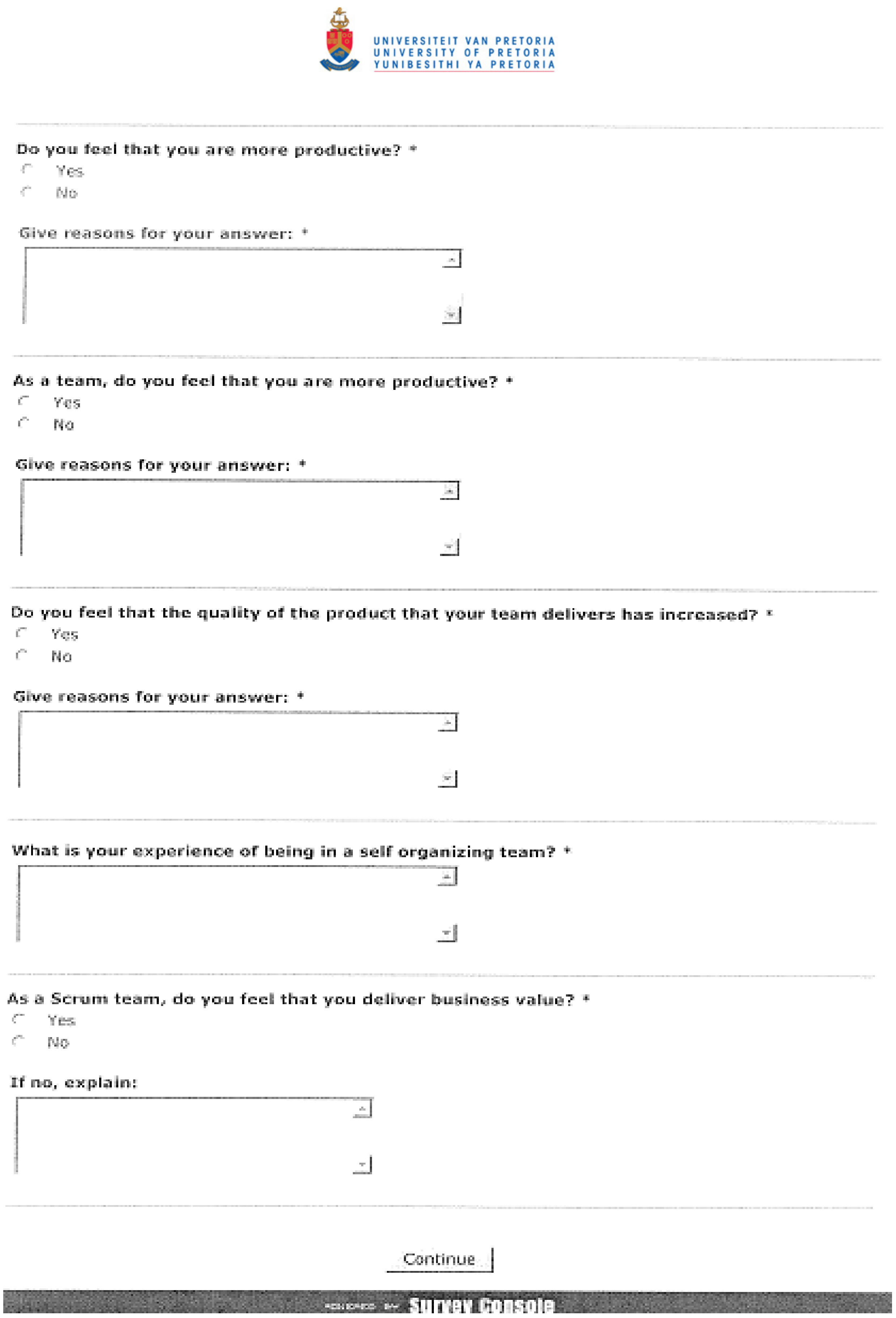
- Yes
- No

Give reasons for your answer: *

Are you comfortable with multiplicity in roles? (developers testing, etc) *

- Yes
- No

Give reasons for your answer: *


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Do you feel that you are more productive? *

Yes

No

Give reasons for your answer: *

As a team, do you feel that you are more productive? *

Yes

No

Give reasons for your answer: *

Do you feel that the quality of the product that your team delivers has increased? *

Yes

No

Give reasons for your answer: *

What is your experience of being in a self organizing team? *

As a Scrum team, do you feel that you deliver business value? *

Yes

No

If no, explain:

Appendix C- External evaluation

BI Data-Mart team

Some of the good

While the team has difficulty with delivering predictably and incrementally, there is a strong desire amongst the team members to identify and fix the problems that are causing them pain. It has been mentioned to me that regardless of what the Sprint history suggests that the business is reasonably satisfied with this team's output.

Potential areas of improvement

My initial meeting with the team was to be their Sprint Review and Sprint Retrospective. These sessions did not happen. It seemed the team was divided on whether or not to hold the Sprint boundary events since there were some items that were liable to hold up the release if they weren't given attention. Not only this, but the boundary events were already three days late. The team has not yet found a way to break down backlog items into small enough chunks to make predictable incremental delivery (at least within a 2 week period) a reality. The team offered some reasons as to why this is currently a problem for them:

- ETL process “just takes a long time”.
- Future-proofing design is slowing progress.
- Technical debt is slowing progress.
- Maintenance and operational work slows down ETL work.
- Environments are slow, and loading data takes a long time.
- • Although an effort has recently been started to produce automated regression tests, the coverage of these tests is still low.

Massive fluctuations can be seen on the team's velocity history. The team feels that most of their previous estimates are worthless, and this may be the case. Another observation is that apart from “Data Cubes” backlog items, the work that the team is doing, is essentially work for other development teams, meaning that there is no direct business value that the team is delivering.

This creates dependencies that need to be managed between teams.

The current approach is to stagger dependant work a Sprint ahead of the consuming teams. This approach potentially increases the time between concept and cash, and implies a specific and rigid way of working e.g. “finish the data transforms and then we can start on the feature that consumes that data”. This also has the effect of separating the problem solving skills of the people working on the actual business solution and essentially keeps all work in process until the business value is actually delivered (or at least in a shippable state). These are some of the arguments for having cross-functional, business-value driven teams, rather than splitting teams by discipline.

It would seem that the challenges and problems that Scrum makes visible are numerous, and the team doesn't really know where to start with solving them. Instead they simply seem to be in the mode of “let's just do as we did before”. The problem with this approach is that improvements will happen by luck rather than concious thought and action, and the confidence in release plans will no doubt remain dubious at best.

Where to from here?

This team needs to establish an agreed and currently achievable definition of “Done”, understand the shortfalls and agree on a plan to improve these over time. Current level of “Done” directly impacts the reliability of estimates and commitments made by the team.

Build a quality product backlog with PO and Team.

Produce and maintain a Product Burndown chart in order to track progress.

Highlight and improve the use of sprint boundary events e.g. planning, review and retrospectives.

Coach team, PO and ScrumMaster through a couple of Sprints to emphasise their roles in the process and help the team as a whole to improve.

Institute Agile metrics in order to track improvements made.

Review dependencies on other teams, cross-team impediments and team structure; is staggering working? Would it be better to combine teams e.g. Communications engine and Datamart? Are they truly working from different backlogs?

Establish regular estimation sessions to continually refine the product backlog and to extend the strategic planning ability of the team.

Review the sprint length. Does a two week Sprint cycle make sense for this team?