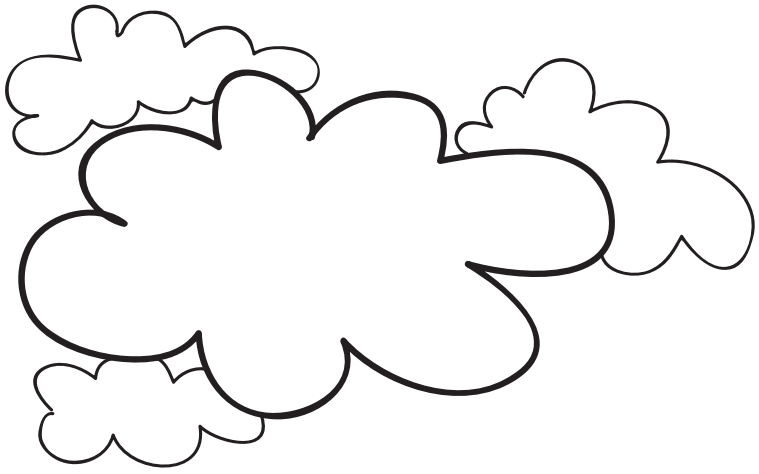
Vision Development

Balancing software innovation and business models.



Department of Computer Science
Aalborg University
January 15th 2015

AALBORG UNIVERSITY
STUDENT REPORT

Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg East
http://www.cs.aau.dk

Title:

Vision Development — Balancing software innovation and business models.

Topic:

Software Development

Project Term:
P10, Fall 2014

Project Group:
is101e14

Students:
Anders Eiler

Supervisor:
Ivan Aaen

Copies: 3

Pages: 109

Finished: January 15th, 2015.

Synopsis:

Vision Development is a methodology that enables everybody involved in a software development project to make intelligent and justified decisions to ensure, that the project is always heading in the desired direction – even if the project should go in a new direction during the development. By using SCRUM and Essence as theoretical foundation, vision development adds tools, methods and questions to the known process that facilitates a development environment with focus on the impact of our decisions, both on the product, but also on the customers business.

ABSTRACT

During the development of the Cloud Apps prototype in the second report on the topic, the development team found it difficult to ensure, that what they were doing was actually what the customer wanted. This report contributes with a new development process, named Vision Development. By using vision development, the Team, SCRUM Master, Product Owner and Customer had a methodology that helped them steer the project, using a common product vision as guideline. The purpose is to ensure, that the Team is creating the software, that the customer wants, and to ensure that the software the customer think he wants, is what he actually needs.

This report describes the Vision Development methodology and presents a case study in which the methodology was applied. Vision Development builds on top of SCRUM and Essence by adding different tools and methods to the known development process. It implements the Business Model Canvas by Osterwalder and Pigneur to understand how the product fits into the customer's business, and it implements Business Patterns by Gassmann, Frankenberger & Csik to understand how the business fits into the market and what central elements and processes it consists of. By knowing that, the customer and product owners are able to create a product vision that guides the project.

It is described how to create a product vision and what to do, when it needs to be revised to include changes, new features or new requirements. Changes to the vision can emerge both internally in the team and externally from the customer. By also building the methodology on Essence, software innovation is an integrated part of the development process, facilitating an environment where investigation and testing of new ideas are encouraged. Two ways of evaluating the ideas are presented with the purpose of making sure, that new ideas are desirable and feasible before implemented. Changes to the vision can also emerge externally, e.g. if the customer wants the product to include new features or otherwise be modified.

In both cases, Vision Development serves the purpose of evaluating if a diversion from the existing vision is desirable or not, by holding it against the Business Model and Business Pattern, so that the full impact of any change is understood. If the changes are desirable, it is described how the product vision can be modified and the changes can be added to the project. The artifact that this report presents, is a product vision that is revised after every sprint to ensure it describes the current wishes and requirements. The product vision is shared among all parties involved in the project to ensure that everybody is working towards the same goal. A goal that is always aligned with the customer's wishes and needs, and one where the products impact

on the customers business is understood. This should result in better products that delivers exactly what the customers needs.

RESUME

Under udviklingen af prototypen i Cloud Apps del 2, havde teamet svært ved at sikre sig, at det, som de udviklede, faktisk også var det, som kunden ønskede. Denne rapport præsenterer en ny udviklingsproces kaldet Vision Development, og præsenterer et case study hvor Vision Development bliver anvendt. Ved at bruge Vision Development havde teamet, SCRUM Master, Product Owner og kunden en metodologi, som hjalp dem med at styre projektet gennem en fælles produktvision som mål at styre efter. Formålet er at sikre, at teamet laver den software, som kunden ønsker, men samtidig også, at den software som kunden tror, at han ønsker, faktisk også er den, som han skal bruge.

Vision Development bygger på SCRUM og Essence som fundament ved at tilføje forskellige metoder og værktøjer til den kendte udviklingsproces. Det anvender også Business Model Canvas af Osterwalder og Pigneur for at give forståelse for, hvordan produktet passer ind i kundens forretning. Derudover anvendes Business Patterns af Gassmann, Frankenberg & Csik, for at give forståelse for, hvordan kundens forretning passer ind i markedet, samt hvilke centrale processer den består af. Med denne viden er kunden og product owner i stand til at lave en produktvision som projektet kan styre efter.

Denne rapport beskriver hvordan den første produktvision laves, og hvad man gør, når det er nødvendigt at revidere denne for at inkludere ændringer, nye features eller krav. Et behov for ændringer i produktvisionen kan opstå både internt fra teamet og externt fra kunden. Ved også at basere metodologien på Essence er software innovation en naturlig integreret del af udviklingsprocessen, som faciliterer et miljø, hvor der opmuntres til, at nye ideer undersøges og testes. Rapporten beskriver to måder at evaluere nye ideer for at sikre, at disse er ønskværdige og kan betale sig, før end de implementeres. Ændringer til produktvisionen kan ligeledes opstå eksternt, eksempelvis hvis kunden ønsker at produktet skal inkludere nye features eller på anden vis modificeres.

I begge tilfælde tjener Vision Development formålet at evaluere hvorvidt en ændring i visionen ønskes gennemført eller ej, ved at holde ændringen op imod forretningsmodellen (business model canvas) og forretningsmønstrerne (business patterns), således den fulde konsekvens af en implementering af ændringern forståes. Hvis ændringen ønskes, er det beskrevet, hvorledes produktvisionen kan blive tilpasset og ændringerne tilføjet til projektet. Resultatet er en produktvision der revideres efter hvert Sprint, for at sikre, at den til enhver

tid beskriver de aktuelle ønsker og behov. Produktvisionen er fælles blandt alle parter involveret i projektet, så alle kan bruge produktvisionen som et fælles mål at styre efter. Dette skal resultere i levering af bedre produkter, som leverer præcis hvad kunden skal bruge.

PREFACE

This report is the third in a series of three reports that follows the Cloud Apps project from an idea in the first report to a prototype in the second and finally vision development in this report.

I would like to thank my supervisor Ivan Aaen for his help and guidance throughout the project. I would also like to thank my friends and colleagues at Meebox for their help and assistance during the development. Finally, I would like to thank Esben Pilgaard Møller & Christian Iversen for helping me with structuring and proof reading the report, and Lenea Lind for the graphical illustrations on the front page and in the report.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AaaS Application as a Service

API Application Programming Interface

DBMS Database Management System

FTP File Transfer Protocol

GUI Graphical User Interface

IaaS Infrastructure as a Service

NAF Novelty, Attractiveness and Feasibility

PMI Plus, Minus and Interesting

1 INTRODUCTION

This project is based on a proposal from Meebox ApS. The idea is to create a new platform for Web-hosting, which is the market in which Meebox operates.

This report is the third and final about the Cloud Apps project. Cloud Apps Report One [11] and Two [13] cover different aspects of the project, including the business model, software innovation, system design, implementation of a proof-of-concept and evaluation of the chosen technologies, testing, development methods, the proof-of-concept system and how it all fits together with the original idea. Those two reports cover a twelve month period of work, which succeeds a period of idea generation in Meebox. It has been more than 18 months since this project was started. One of the things that concerned me during Report Two, was if we were still heading in the right direction. Are we still creating the product that the end customers want? Is the product still up to date with what Meebox wants? Are we even implementing this in the right way, using the right techniques and technologies? None of the tools that we have been using, including the Business Model Generation [24], Essence [2], Object Oriented Analysis & Design [21] and development process methods like SCRUM [32] & XP [36] has been able to answer those questions. Although SCRUM is capable of handling changes in the requirements during the development of the product, those changes usually emerge externally [31]. The Cloud Apps project needed a way to handle both internal and external changes to the product, and a way of ensuring, that the accepted changes were desirable to both the product but also Meebox' business.

While I continue the implementation of this project in Meebox with its engineers and developers, I will use this report to investigate how to make sure that the project still has the right vision and is heading in the right direction. I want to investigate how we can make sure that what we develop is still what the customers want, and how to do this when you know there is a risk that the goals may change along the way. Software innovation encourages all parties to be innovative and to think of new and better solutions to the challenges they are faced with. How do we know that the innovation-process has not gone down a path that deviates so much from the original idea, that the product will deviate from what the customer needs? The experi-

ences from Report One and Two and the product will be used as the subject for the investigation.

1.1 PRODUCT VISION

The idea of having a project vision is largely based on the Project view from the development methodology, Essence [2]. It is related to project management, but it is not the same as traditional project management. It is not about handling resources, costs, task planning and assignment etc. It is about facilitating an innovation process that focuses on stimulating creativity throughout the entire project, rather than only at the beginning. Instead of trying to eliminate uncertainty and ambiguity as traditional project management would, innovation management works with the uncertainty and tries to convert it into new ideas, inspiration and possibilities, while still steering the project in the right direction.

In Essence, the customer's wishes and requirements are represented as Challenges. The first product vision is usually a representation of the challenges that should be solved (see Section 2.3 for an explanation of challenges). The vision might change as the team gains new insight and knowledge about the project, challenges and project domain. If a project is locked by a list of requirements given at the beginning, chances are that, even though the product may comply with the list of requirements, it may not be what the customer actually wanted[1]. If a project uses agile methodologies during development and planning, embracing collaboration with the customer, self-organizing teams and short sprints with lots of presentation, testing and feedback from the customer, the project may be heading in the right direction and deliver what the customer think he wants, but it misses out on the opportunities to explore new technologies and solutions that ultimately creates a better product[2]. Software Innovation encourages the team to seek out new possibilities and solutions to the challenges, to deliver what the customer *needs*, but at the risk of getting lost down the wrong path along the way. A clear project vision to lead the way, combined with tools and methods to ensure, that the project stays on the right track, and methods to get back on the right track if it is not, could solve the problem. This should ultimately lead to a better product, giving the customer what they actually need instead of what they think they want.

1 Traditional development and project management is explained in further detail in Chapter 2

2 Agile development methodologies are explained in further detail in Chapter 2

1.2 CLOUD APPS

The section will explain the concept of Cloud Apps, and give a brief recap of the first two reports. It will also highlight what problems we saw in those reports, in regard to project management and keeping the vision straight. Parts of this section will therefore be largely based on Cloud Apps Report 1 [11] and 2 [13]

> " *A Cloud App is an Application that is hosted in the Cloud in a self-contained isolated environment. It is automatically provisioned upon request and it automatically scales its resources to handle the load at any given time. Furthermore, it is automatically updated and maintained.* "
>
> — Definition of Cloud Apps, Cloud Apps Report 1 p. 8

The quote above states what Cloud Apps is. There are a number of key features that makes it new and unique. The following sections will discuss those, looking at both the original idea and what the first two reports found when trying to realize them.

1.2.1 *Application as a Service*

Traditional web hosting usually provides IaaS *[Infrastructure as a Service]* that gives the customer a number of services such as a web-server (e.g. Apache or nginx), DBMS *[Database Management System]*, FTP *[File Transfer Protocol]* server, mail-server etc. that he can use to host the wished web application. It can be a technically demanding task for the end user to manage all these services and get the web application up and running. Using traditional web hosting gives a lot of freedom, in that the customer can install and run almost any web application he wants.

Cloud Apps offers those same applications as a service. Instead of purchasing a hosting account where the customer can install his preferred application, he purchases the application he wants and focus on the application and nothing else. All the systems and services required to run the web application are handled by the Cloud Apps system. The customer chooses his application and purchases that – therefore it is AaaS *[Application as a Service]*.

There are a number of key features that makes Cloud Apps more than a the usual application installer. They will be described in the following sections.

1.2.2 *Hosted "in the Cloud"*

Every Cloud App instance will be hosted in an isolated container. This container can be anything from a jailed environment to a virtual private server or a whole physical machine. Cloud Apps Report 1 and 2 defined that instances will have their own virtual server in a Cloud Environment. It is designed to be platform independent. Cloud App instances can be provisioned in any Cloud Environment that is supported by the system. That could be Amazon[3], Rackspace[4], Digital Ocean[5] or Meebox'[6] own.

There are a number of reasons why it was decided that Cloud Apps should be both hosted in a Cloud Environment and be platform independent:

- Data Integrity. To ensure that each customer's data is always isolated from other customers, each Cloud App instance will be hosted in isolated containers.
- Performance. To ensure that one Cloud App instance cannot affect the performance of others, they will be hosted in isolated containers.
- Easy scaling. By hosting each instance in isolated containers, it is easy to control the resource allocation to each instance, making up- and downscaling of the allocated resources easy.
- Market positioning. By integrating with multiple Cloud Hosting providers, new instances can be provisioned in multiple locations at different prices, allowing the customer to customize to his needs.

During the development of the prototype of the system in Report two [13], it was found that platform independence can be achieved by having an extra abstraction layer between the Cloud App instances and hosting providers. This can be done by normalizing the required features such as create, edit, delete, start, stop, scale etc., and have specific implementations for the available hosting provider's Cloud Environments.

1.2.3 *Automation*

One of the key features of Cloud Apps is its ability to automate a number of the most important and tedious jobs concerning application management, such as provisioning of new instances, scaling up

3 http://aws.amazon.com/ec2/
4 http://www.rackspace.com/cloud/
5 https://www.digitalocean.com/
6 http://meebox.net/en/cloud-server/

and down to serve higher or lower load and updates & maintenance. Cloud App's way to automate all three areas are discussed in the following subsections.

1.2.3.1 *Provisioning*

There are a number of steps that must be completed to create a new Cloud App instance:

- Provision a new server.
- Install and configure an operating system.
- Install and configure the required services and dependencies.
- Install and configure the chosen application.

Each of the four requires different tools and technologies. Provisioning of new servers is handled by the Hosting Providers. The hosting providers typically also installs the operating system. Report 1 [11] did a survey of three configuration management systems and found that MetaConfig [17] was the better choice.MetaConfig can be configured to both install and configure Linux applications, making it useful to install and configure both the required services and dependencies and then the chosen application.

One of the important elements in the business model is automation. Every Cloud Apps instance should not be configured manually by an employee, as this would be too expensive, and at the same time it should be ready to be used by the customer within minutes from ordering.

1.2.3.2 *Scaling*

Being successful should be an advantage, not an unwanted challenge. If a service gets too many incoming requests at the same time, it may be slower to respond or fail to respond. One can of course choose to have a permanent resource allocation that can handle load spikes, but that will often be too expensive. Instead Cloud Apps should offer automatic scaling of resources.

Report 1 [11] discussed two types of scaling (scaling in/out and scaling up/down [22] [35]) that allows a Cloud App instance to handle load spikes without affecting the performance. By surveiling each individual Cloud App instance, it is possible to identify when it is necessary to allocate more resource to it, to ensure that the application responds and behaves as required. The same idea goes for scaling down, which is useful when the load is low but the customers wants to save money on not having unnecessary many resources allocated.

1.2.3.3 *Updates and maintenance*

Cloud Apps should allow the customer to focus on his application and nothing else. However, ongoing updates, maintenance etc. is still required to keep the system running. The operating system, required services & dependencies and often the application itself needs to be updated on a regular basis to implement security improvements, bug fixes etc. Cloud Apps will use its configuration management system to handle automatic updates and maintenance of all required systems, making sure that they are always up-to-date and that the customer should not be bothered with it.

1.2.4 *Business Model*

The business model from Report 1 [11] was created to help identify who the customers are, what they want and how it can be provided to them. It also shows how Cloud Apps fits into Meebox' existing business and what it can bring of new opportunities to the business. What are the products strengths, weaknesses, opportunities and threats? The business model is based on the Business Model Generation [24] and Software Innovation [1].

Cloud Apps was identified as intrapreneurship, meaning it is innovation within an existing organization. Cloud Apps is based on a multi sided business model with three different, but dependent, customers. They are:

- End Customer. The End Customer is the one purchasing the Cloud App instances and deploying their systems with Cloud App instances. They need applications and cloud environments in which the Cloud App instances can be provisioned.
- Application Developers. Application Developers develop new applications that can be provisioned as Cloud App instances. They need end customers to purchase the Cloud Apps and cloud environments in which they can be provisioned.
- Hosting Providers. Hosting Providers offers their Cloud Environment to Cloud Apps so new instances can be deployed there. They need applications to deploy and end customers to purchase the Cloud App instances.

The End Customer will purchase new Cloud App instances in the Cloud Apps Marketplace, an online store that displays the available applications that can be purchased as Cloud Apps. Examples of such applications are:

- Web Services such as Content Management Systems (e.g. Wordpress, Joomla, Drupal, Sitecore, Umbraco etc.), Shopping sys-

tems (e.g. Prestashop, Magento etc.) or other web applications/frameworks.

- Software Services like web servers, database servers, mail servers etc.
- Infrastructure to support Big Data, Data Warehousing or complex computational tasks based on e.g. Apache Hadoop[7].

The End Customer will be able to choose from the available Cloud Apps in the Cloud App Marketplace, choose the preferred hosting solution from one of the hosting providers and then provision the Cloud App instance. At that point, the customer will have an application as a service that is ready to use.

The sections above describe the idea of Cloud Apps and shows some of the findings in the first two reports of this project. Report 1 was focusing on the business model and system design. It identified Cloud Apps as a product with a multi-sided business [8] that will fit into Meebox' existing business. It also presented a draft for a system design that would later be implemented as a prototype. Report 2 designed, implemented and tested the prototype, as it focused the implementation of a proof-of-concept version of Cloud Apps using the system design and business model from Report 1. The proof-of-concept was tested by a number of chosen test subjects, who found that the product was heading in the right direction. It delivered what they expected of a proof-of-concept.

We, the development team at Meebox, may have been lucky this time, that the proof-of-concept that was presented more than a year after the original plans were drawn, still delivered what the test-subjects (End Customer) expected of it, taking time, technology and other potential changes into consideration. The question is not how to seek that same luck next time. The question is how to ensure, that the next project also delivers what the customer wants.

1.3 NAVIGATION

During the Cloud Apps project, I have been working with complexity on three different levels.

- Product. The complexity of the product is often a technical complexity, discussing specific details about the product itself. How do we design this system? What technologies are available and do they offer the features we need to implement the product? These are questions that you would ask during a sprint in iterative development, that solves problems close to the product.

7 http://hadoop.apache.org/

8 Multi-sided business models are discussed in Chapter 4

- Project. The project's business model has a broader view than the product, as it takes the customers business into consideration. It looks at whether what we are doing fits into the existing business model for the product. During sprint reviews or sprint planning in iterative development, you would ask questions like: Does the product we are developing fit into our business model? Is it what the customer wants? Something may have come up during the last sprint that has changed the vision of the project, but still fits within the business model. That would affect what we are doing on the product-level.

- Business. The business patterns attached to the business forms the grand overview of the business. What type of business patterns does the business use? Is this project following those patterns, or is it pushing the business towards other patterns? Is that a good or bad transformation? How do we determine if that is a good or bad transition? This view is applied between projects. Before a project is started the business has an idea of what direction the project will take the business. After a project finishes, it can be evaluated in which direction it actually took the business. Is it the same as expected? Decisions made at this level affects both the business model and the product being developed.

My mother and step-father owns a 28 feet ship named "Augusta 2". Every summer she is taken out from her harbor in Aalborg, northern Jutland, to sail down the eastern coast of Jutland. The tasks that my step-farther, John, faces as Captain of "Augusta 2" are in many ways the same as a company developing software faces, when looking at it as individual tasks from different levels of abstraction. When sailing on open sea, you have a starting point and a destination you wish to reach. The journey is affected by the wind, the weather and a number of other factors. The vision is clear: We want to get to our destination. The vision is broken down into smaller components that are easier to handle. Such components could be to get out of the harbor in Aalborg, to sail down Limfjorden towards its merging with the Kattegat sea. Then sailing down the coast, closing in on the destination before finally reaching it and laying in at the destination harbor. Each of those components can be seen as seen as individual projects with different requirements and part goals, all relative to the larger components (models), that are related to the overall vision.

When sailing with sails instead of engines, you rely on the wind to drive the ship forward. Sometimes, the wind is not always with you, and the compass points you up against the wind. That is a problem you have to solve here and now, in order to continue the journey and follow your vision. This can be seen as a sprint in an iterative development project. In this situation, the Captain would start to zig-zag

the boat up against the wind to catch it from the sides, and use that force to drive the ship forwards. In doing so, he must also ensure that he stays clear of land and does not get off of the course. That is an external factor that affects the current project of getting out of Limfjorden and into the Kattegat sea. Another external factor could be my mother (who can be seen as product owner in this context) who suddenly decides to change directions and set sails towards Samsoe. That changes the vision, which in turn also affects both the short- and long term projects. The Captain now has to navigate a different route that faces other challenges. He has to embrace the change, adapt to it and solve the new tasks at hand.

The journey is the business. The individual components in the journey are projects. The small challenges, like zig zagging, are sprints in those projects. It is important to place the horizon at the right level to see what is important in the given situation, but still understand how that affects the immediate future and abstraction layer just above. As the Captain of "Augusta 2", you do not have to understand why my mother suddenly wants to go to Samsoe. You just have to get the ship there. As product owner, you do not have to understand how the Captain gets the ship to the destination, or how it is possible to sail up against the wind. You just trust the Captain to do his job, lean back and enjoy the journey. In both this situation and software development projects, it is important that the product owner trusts the team to do their job and not try to micro-manage, e.g. by telling the captain in what direction he should sail. That could prevent the captain from doing his job, or at least make it more difficult than necessary.

This project is about understanding what is important right now, while still being aware of how the decisions I make now affects the immediate future, the rest of the project and the rest of the business. In the two previous reports I have been working with the Product abstraction (in Part 2) and the Project layer specific to the product (in Part 1). This report will work with understanding the third layer, the Business. It will identify one or more patterns that this business matches, and identify whether this project has transformed the business to match other patterns. And if it has, determine whether it was a desirable transition.

1.4 RESEARCH QUESTIONS

There are a number of questions that need to be answered before we are able to make an intelligent decision on whether a change in the vision is desirable or not. After those are answered, we need a new way of working that embraces this way of thinking, where we try to have a deeper understanding of the business as a whole, while we still allow

for software innovation to happen. The research questions originates from the work I did and the experiences I got while working with this in the first and second report. It is centered around identifying and handling changes in the vision. They address three different areas, namely vision handling, change impact and available/relevant tools and methods. The research questions are:

- Vision Change
  - How do we know if the vision has changed?
  - How do we handle a vision change?
- Change Impact
  - How can a change in the vision affect the Business Model?
  - How can a change in the vision affect the Business Patterns?
  - How can a change in the Business Model affect the vision?
  - How can a change in the Business Patterns affect the vision?
- Available/relevant tools and methods
  - What tools and methods are required to evaluate a vision and idea?
  - How can we facilitate a development phase that can embrace changes to the vision?
  - How can vision development include software innovation, and how do they affect each other?

I will answer these questions during the last two parts of this report.By answering those questions I will have the required knowledge to expand on the validity of the hypothesis.

1.4.1 *Hypothesis*

The hypotheses that I will be working with in this report is:

Adapting a known development method to include vision development will enable the customer, developers, product owners and managers to actively see, understand and work with a product vision, and use it as a common goal to aim for during the development of the product, to ensure that it is heading in the desired direction. It will also identify both inner and outer changes to the vision, and based on that, enable the customer and product owner to make intelligent decisions on whether the change in the vision is desirable or not.

I want to use this report to create something that I have needed myself, and that I believe is needed elsewhere too: A development methodology including vision development. A methodology that enables everyone working on a project to see both short-term and long-term goals of a product or project, and the direction in which they are going, but also make them aware of changes, their impact and when it is necessary to implement them. By doing that, I believe more projects will result in better products that delivers what the customer needs and expects.

APPROACH

This report is split into thee parts. This is the last section of the first chapter, containing the introduction, project presentation and the hypothesis. Following the introduction is the three parts in the report. Part I, background, discusses relevant knowledge and literature that is needed to answer the research questions and hypothesis. Part II, methodology, contains the methodology and a case study in which vision development was applied. Here the knowledge from Part I is put into practice, and it is described how I imagine the hypothesis can be answered with a practical approach to vision development. Finally the evaluation in Part III contains a discussion of the findings in this report and end with the conclusion.

Part I

BACKGROUND

The purpose of this Part of the report is to cover the necessary topics to discuss development processes, visions and software innovation and how these concepts can be combined as described in the hypothesis. The development processes, vision and software innovation topics will each be discussed based on relevant literature and the Cloud Apps project project. Examples of how they were each used in this project, and what effect they had, will be shown too.

2 DEVELOPMENT METHODOLOGIES

Different development methodologies in software projects is one of the more covered topics within Information Systems. Choosing the right development methodology has always been important, because everyone wants to make sure, that a project delivers the best result possible. Choosing the right development method for a project is related to the vision and the goals that a development team are targeting when creating new software. What methods and methodologies should be applied to best facilitate both software innovation and keeping the project on the right track at all times? That question is answered in this chapter.

This chapter will explain three development methodologies: Traditional development (also known as the Waterfall model), Iterative development (also known as Agile development) and Software Innovation (based on Essence). Their common grounds and differences will be discussed and they will each be held against the Cloud Apps project to estimate what would have happened, had the project been using another development methodology. An existing and known development process is the foundation of vision development. This chapter serves the purpose of discussing three development processes, comparing them to each other and the Cloud Apps project, and discuss what would be an appropriate foundation for vision development.

2.1 TRADITIONAL DEVELOPMENT

Traditional development takes a linear approach to software development in which a project usually consist of a sequence of events like:

- Research – Gather all the required information, documents and other knowledge required to develop the software.
- Design – Based on the research, create a system design.
- Implementation – Implement the system that has been designed.
- Testing – Perform substantial testing to ensure all pieces of the software are working.
- User testing – Test the software with the users and ensure that everything is working.
- Bug fixing – Fix any bugs that were found during development.

- Delivery – Deliver the finalized software to the customer.

These steps are usually performed sequentially and not overlapping as they are each assessed and accepted by a third party, like a manager, before the next phase of the project can begin.

The traditional methodology has a number of advantages that makes it preferable in some projects. The developers and customers can agree early on the scope and requirements of the project, thus early on agreeing on what must be delivered from the project. During the project it is easy for the customer to track the progress and see the status of the project. Because the project follows a waterfall-like model, the individuals working on the project can go to/from the project as required. E.g. system testers can begin to prepare the test scripts as soon as the design is done, because they know that the implementation team will create the system according to those same design documents. Finally this methodology opens for a very thorough system design, because all knowledge is present and the full scope of the project is known already at the design phase. This can help avoid "hacks" where developers implement small fixes here and there to resolve unforeseen issues.

There are two major issues with the traditional development methodology. The first is the lack of support for changing requirements. It can be very difficult to gather the necessary information to document all requirements in a way that makes sense to the customer before the design and implementation starts. If the customer does not understand the product at this time, chances are that the final product will not satisfy them. The second issue is the strict sequence of phases in the project. If the customer is not satisfied or if the customers requirements change during the project, it can be very difficult - if not impossible - to adapt to these changes. Often the project would have to be completed, and a new project for updates or maintenance will be created to handle the new requirements from the customers.

From the beginning, Cloud Apps has been focused on creating something new, based on the existing technologies and requirements from the customers. If the project had been using a traditional development methodology, we believe that the project would have failed to deliver a product matching the requirements. The primary reason for that, is that the requirements are constantly changing. For example, if new technology becomes available that the system has to support, or if the industry changes and the Cloud Apps system has to be adapted to these new changes. Because it is expected that the requirements to the Cloud Apps system will change multiple times during the time it will take to develop the system, we believe that a traditional develop-

ment is not the best choice for this project or the Vision Development methodology.

2.2 ITERATIVE DEVELOPMENT

Iterative development is often directly associated with the Agile Methodology, as is the case with this project. It is a team-based approach to software development that emphasizes rapid delivery of working code. Instead of creating tasks and scheduling them, the project is split into smaller iterations called "sprints". Each sprint can be looked upon as a mini-version of the traditional methodology. A sprint usually lasts between two and four weeks depending on the project type and available staff, and contains the full process of research, design, implementation and testing. After each sprint, the development team must deliver working software that can be shown to the rest of the team and the product owner. If the scope of a sprint is larger than what can be realized within the agreed time frame of a sprint, instead of making the sprint longer, the scope is made smaller and everything that is not done is pushed to future sprints. When there are too many tasks in a sprint, the most important tasks are prioritized thus delaying less important tasks.

Because every sprint delivers working software, the customer can continuously review and evaluate it and compare it to his expectations. The customer usually provides a "Project owner" that oversees the new software and ensures that it is, what he is expecting.

This methodology also has a number of advantages, making it preferable in some projects. The customer has frequent and unlimited access to see the work that is being delivered, and is therefore able to make decisions and apply changes throughout the project, thus the methodology allows for change. The product owner determines the priority of features, and the team therefore gets a better understanding of what is important to the customer. Agile development can adapt the final product to changing circumstances during the development, e.g. will it be possible to deliver a basic version of a piece of software, if time-to-market is a concern.

On the other hand there are some disadvantages when using this methodology. Not all customers are able to provide a project owner. Because the development is split into several iterations, it is possible that some features cannot be delivered within the given time frame, thus they must be developed in a later sprint. This can cause the delivery to be delayed. The iterative nature of this methodology can cause that too much focus is put in the individual sprints, and not enough on the final product that is created from the results of the individual sprints. The larger the project is, the greater the risk that this can hap-

pen.

There are a number of variations of iterative development. The most popular ones are SCRUM [32], Extreme Programming [36] and Unified Process [4]. They are specializations of iterative development, meaning they each define the processes in greater detail, such as how weekly meetings should be held, using pair programming etc. Though it is relevant which method is being used in a project, it is only relevant after the decision to use an iterative development methodology is made in the first place, and they will therefore not be explained in greater detail here.

Had I chosen to use an iterative approach during the development of the Cloud Apps project, it would have been more likely to succeed compared to if I had chosen a traditional development process. An iterative approach would allow for change during the project, and support the need to be able to respond to changing circumstances. Choosing an iterative process would also allow the development team to get frequent feedback from the customer, which is an advantage in this case, because the project is navigating in unknown waters. The problem using one of the existing iterative development processes is that it does not actively encourage innovation to the extend that is needed.SCRUM can handle changing circumstances and modifications to the back log, but usually those changes has to emerge externally [31]. Innovation and new ideas that emerge internally is a key part of the Cloud Apps project and vision development, and therefore a classic iterative development process was not chosen as the foundation of vision development either.

2.3 SOFTWARE INNOVATION

The idea of using innovation in software development is based on the methodology named Essence [2]. Essence builds on SCRUM's iterative approach and encourages more focus on software innovation during the development process with its values, views and roles. This section will briefly describe the idea behind Essence, how it was used in this project, and why I believe it was the right choice.

Essence is based on four values that serve as guiding principles:

- Reflection over requirements. Both traditional and agile development agree to comply with customer requirements. Instead of just complying with the requirements, software innovation asks: How can development teams deliver high-value solutions? Because innovation is about learning and discovering and exploring new options and possibilities, focus should be on re-

flecting over these new findings rather than simply complying with customer requirements.

- Affordance over solution. How can a project deliver the solution that the customer wants? Both traditional and agile development sees development as being about delivering a solution that matches the customers needs. Instead, new options and possibilities are discovered during the development of the product as the result of experiments being performed. Both the application and domain of the project may develop as new technologies and possibilities are discovered.
- Vision over assignments. When an idea becomes one or more tasks that are assigned to a developer, the innovative process is essentially over. At this point the developer's mind switches lanes from idea-generation and exploration to problem-solving and implementation. Project management must be looked upon as idea/vision generation and maturation. The scope and goal of a project must be able to be redefined as the project unfolds, and the vision along with them used to inspire and guide the development team in the right direction.
- Facilitation over structuration. Traditional and agile development uses different methods to structure the teams. Traditional development uses different tools and processes to ensure that similar tasks are performed in similar ways, while agile development develops the processes step-by-step, based on the team and personal competencies. Common for both are that they focus on structuration. Instead of trying to structure the development with predefined processes and tools, we should offer flexible support for learning and developing new solutions. Instead of focusing on the process itself, we must focus on how to facilitate learning, exploration and discovery processes.

Those values are the foundation of Essence and originate from [2]. Essence has three main components that will be described in the following sections. Those are Views, Roles and Visions.

2.3.1 *Views*

Essence has four different views on a project. The four views exists to introduce order into an otherwise chaotic process of idea generation. The views are inspired by the 4P model (product, project, process, people) from Software Engineering (Pressman, 2005; Bernstein & Yuhas [7]) and Tidd, Bessant & Pavitt's model (product, position, process, paradigm) from innovation (Tidd et al., 2005). The four views in Essence express important details in the process while still maintaining an overview. The four views are:

- Paradigm – This view sees the solution from the outside, e.g. as the customer, a user or an external system component that needs to interface with the solution. New ideas that emerge when expanding on ideas or investigating challenges in detail are often put here until they have been confirmed as potential to be implemented. Refers back to the "Reflection over requirements" value.
- Product – This view sees the solution from an engineering or technical point of view and focuses on how to build the solution. It provides an overview of the technical aspects and allows for experimentation with new technologies, architectures and system designs. Refers back to the "Affordance over solution" value.
- Project – This view sees the project from a management perspective. This view develops and maintains the overall project vision based on the emerging of ideas, how they are broken down into smaller components and finally individual tasks. This view will be the primary focus for the majority of this report, since this is where the project vision is handled. Refers back to the "Vision over assignments" value.
- Process – This view gives an overview of the methods, tools and techniques that may be of use when generating and maturing ideas. It is a box that contains everything that does not belong elsewhere, such as idea quality evaluations, research evaluations, cost and resource analysis etc. Refers back to the "Facilitation over structuration" value.

Essence recommends creating four physically separated boards, one for each view, to allow the team to better separate them mentally when working on the different views. Using the four views to discuss, mature and evaluate ideas should allow the team to "tear them apart", and by doing that, letting them evolve and investigate in which direction it goes. Along with using the views for analyzing new ideas, team members can attend different roles to see the product from different points of view.

2.3.2 *Roles*

Essence describes itself as being light on techniques, tools and procedures but heavy on structures. Roles are a key part of the structures in Essence, as it invites the team members to take different responsibilities and points of views in software innovation.The purpose is to have all team members participate in viewing a challenge from different perspectives, which Essence claims opens up for better idea exploration.The four roles in Essence are:

- Child – explores ideas. The child is curious, has no limits and does not know how to say no. Anything is possible and the child's task is to explore, ask questions, make suggestions and see new opportunities in ideas, no matter how unrealistic they may seem to everyone else. Refers back to the "Reflection over requirements" value and Paradigm view.
- Challenger – challenges the idea. He sees the project from the customers perspective. He approves, disapproves and prioritizes ideas as he things they are most important to the project. This is done at the Project view. Refers back to the "Vision over assignments" value and Project view.
- Responder – tries to apply a technological viewpoint to the presented ideas. He has a functional and practical view on things and can suggest how to implement them at the Product view. Refers back to the "Affordance over solution" value and Product view.
- Anchor – is the main facilitator of the project. The captain, so to speak. He supports the working processes in the team and provides the necessary tools and techniques. He ensures that the team is functional and productive. Refers back to the "Facilitation over structuration" value and Process view.

By applying different roles at different points in time when discussing an idea in the context of a specific view, the development team has the opportunity to discuss, expand and explore ideas in a way that may not have been possible using either traditional or agile development.

The third component, the Vision, will be explained in the next chapter where it will be combined with other literature to discuss how it can be used in vision development.

The Cloud Apps project takes existing technologies and bundles them together in a new way to offer additional value to the customers. That is why Essence and software innovation was chosen as the development methodology to be used in the project and as the foundation of the Vision Development methodology. It embraces changes to the product, project or vision as it unfolds. It facilitates exploration and maturation of ideas to transform them from ideas to possibilities or visions.

This Chapter has discussed and compared traditional development, iterative development and software innovation in the context of the Cloud Apps project. From here on, the software innovation development methodology will be referred to when discussing other ele-

ments in relation to the development method used in the Cloud Apps project.

3 VISION

Every product should have a product vision. A clear, well-crafted and well-communicated product vision should permeate everything that you and others working on the project do. It will affect everything concerning the product from the business model and product roadmaps to stories in the backlog and how they are executed. Consider the product vision as the shining beacon from a lighthouse that can be seen from afar. It helps you to be sure that you are moving in the right direction at all times. It ensures that the strategies are aligned and that the team working on the product uses their time creating the right product. If you are working on a product without a clear vision, chances are that you will have a very different view on what the product's vision is than many of your colleagues. If everybody is not working together towards the same goals, how can it then be possible to reach them?

There are different tools and methods to create a vision for a product. The following sections will describe two different tools to create a clear precise vision with the purpose of describing how it has been done in this project and how it can be done in other projects too. A product vision is the most important artifact in vision development, thus it is important to understand how the initial product vision is created, and how it can later be adapted to changes.

3.1 VISION STATEMENT

Platinum Edge has developed a four-step process of creating a vision statement [9]. The process of creating a product vision precedes any other activity in the development life-cycle. Only after the product vision is finalized should the planning and development begin.

The four steps in creating the vision statement are:

3.1.1 *Develop the product objective*

First you need to understand the product. Ask a lot of questions, such as who the users are, how the product is similar to other competitive products, how will it benefit the business developing it etc. The more questions asked, the better an idea you will have about what you are about to create. It is important to involve the product owners and customers in this process to get their view and expectations on the product included into the vision.

3.1.2 *Create a draft vision statement*

One of the more known templates for a product vision (sometimes referred to as a positioning statement [5]) is Geoffrey A. Moore's product vision statement template from his book *Crossing the Chasm*. The template has the following structure:

- **FOR:** the target customer...
- **WHO:** needs...
- **THE:** *product name*...
- **IS A:** *product category*...
- **THAT:** *product benefits, reason to buy*...
- **UNLIKE:** the competitors...
- **OUR PRODUCT:** *differentiates by ...*

Platinum Edge also recommends adding an eighth step:

- This supports the company's strategy to *insert name or description of strategy*

The advantage of using Moore's template is that it is a very simple approach to write a product vision. It is almost the same as an elevator pitch, as it uses the same structure and the vision statement is also brief and can be explained in a minute or two.

3.1.3 *Validate and revise the vision statement*

The next step is to go through the statement a couple of times and revise it with yourself before sharing it with others. Typically, once that is done, you will share it with the product owners, development team and scrum master etc. Use their feedback to see if the vision is clear and easy understandable. If it is not, revise it and make another effort.

Although it is one person that is working with the vision at this time, the vision does not belong to that one person. The vision is a owned by everyone involved in the project and usually managed by the product owners.

The feedback can be anything from changing a word to make the statement more clear, to changing whole sentences or parts because they simply have another view on the product. Especially the latter is a good opportunity to discuss exactly what product you are making, and avoiding wasting time developing code in opposite directions. Repeat this process until the vision statement is easily understood with every party.

3.1.4 *Finalize the vision statement*

The final step is making the vision statement clear to everyone in the team. Print it out and hang it somewhere everyone will notice it, to make it clear to everybody what direction we are heading in.

Using Moore's vision statement, a vision for the Cloud Apps product would be:

- **FOR:** a customer with no technical skills
- **WHO:** needs a scalable web application to support his business
- **THE:** Cloud Apps
- **IS A:** application as a service
- **THAT:** allows the customer to provision the web application he needs without focusing on anything but the web application itself
- **UNLIKE:** other 1-click installers
- **OUR PRODUCT:** has automatic scaling, updates, maintenance and is provisioned in its own isolated container to secure data integrity and performance.
- **THIS SUPPORTS THE COMPANY'S STRATEGY TO:** automate as much as possible and attract a new segment of novice customers.

3.2 VISION BOARD

The Vision Board was originally developed by Roman Pichler in 2012 [26] as a side-project of his attempt to integrate his Product Canvas tool [25] into the online SCRUM management tools JIRA[1] and GreenHopper[2]. During this project he found a need to visualize his product vision to make it clear what he was working on. The vision board template is shown in Figure 1.

The information on the board are overall and not specific. When you are working with your initial vision, you do not have enough information about the users or product to be specific enough to create persona's, design sketches etc. The purpose of the board is to help think through an idea and allow one to share his thoughts with the team or other partners in a way that is easy to communicate. Along

1 https://www.atlassian.com/software/jira
2 https://www.atlassian.com/software/jira/agile

Figure 1.: "Vision Board" by Roman Pichler

side looking at the possibilities, the vision board also helps investigate potential risks about the product.

The vision board consists of the five sections described below.

- **Vision**. This section contains a precise description of the idea, the intentions and the motivations behind it. It is recommended to keep this to a maximum of two sentences. It maps to Essence Project-view 2.3.1 and should be used in that relation to include Software Innovation into the discussion.
- **Target Group**. This describes the markets or market segments the product targets. It should state who will benefit from the product and who its users/customers are. It maps to Essence Paradigm-view 2.3.1 and should be used in that relation to include Software Innovation into the discussion.
- **Needs**. This describes the value propositions of the product. What needs does it fulfill? Why should the customer want to buy this product? It maps to Essence Paradigm-view 2.3.1 and should be used in that relation to include Software Innovation into the discussion.
- **Product**. This section summarizes the key features of the product, describing what it is and what it can do. Keep it short and list the most important ones. It maps to Essence Product-view 2.3.1 and should be used in that relation to include Software Innovation into the discussion.
- **Value**. This section should argue why the business should spend time and money developing the product and what value it will add to the business. It maps to Essence Process-view 2.3.1 and should be used in that relation to include Software Innovation into the discussion.

The Vision Board helps to encapsulate the components that makes up a product vision. It somewhat overlaps with both the Vision Statement in Section 3.1 and the Business Canvas [24], but keeps more focused on the product and the overall view of the product in the context of the business building it. The Vision Board's advantages is that it makes it easy to communicate a fractional product vision to others to get feedback, and it makes it easier to visualize and get a coarse-grained overview of the vision. As with the vision statement, the vision board should be brought back and forth between colleagues, product owners, scrum masters etc. to discuss thoughts, ideas or changes to the vision. That process repeats until everybody finds the vision clear and easily understood.

Using the Vision Board on the Cloud Apps product, it would say:

- **Vision**. Provide web applications as a service with automatic updates, maintenance and scalability.
- **Target Group**. End Customers who needs a web application to support their business, like a web shop, business intelligence etc.
- **Needs**. The customer should only focus on his business and the application itself. Cloud Apps should automatically handle "everything else", being all the technical things that needs to work to provide such a web application.
- **Product**. Automatic provisioning of web applications in a secure environment that guarantees data integrity, performance and scalability.
- **Value**. By freeing the customer from having to focus on the technical aspects of their product, a new market segments of novice customers opens that can suddenly have their business supported by applications in the cloud without having technical knowledge themselves. Therefore they will choose this solution, as it is cheaper than similar solutions, because it is fully automatic.

The original definition from Report 1 of what Cloud Apps is, stated: *A Cloud App is an Application that is hosted in the Cloud in a self-contained isolated environment. It is automatically provisioned upon request, and it automatically scales its resources to handle the load at any given time. Furthermore, it is automatically updated and maintained.* What has up until now been known as a product definition is in fact the product vision that has steered the project so far. Though the vision lacks important parts, such as who the customers are and how the product differs

from competitive products, it is based on that vision that the business model canvas was created and the prototype was created.

The purpose of this chapter was to describe what a vision is, how you can make a vision and how it has previously been used in this project. That knowledge will be used in later chapters when we look at how the vision has changed, and how to determine whether that is a good or a bad thing for the project and the business developing it.

The next chapter will take a higher perspective and look at different business patterns that can be applied to a business or product to identify how it works and how it makes money. It is based on the products that are being sold and how the business operates, both of which are greatly influenced by the vision.

4

BUSINESS PATTERNS

The view you have of a task, a project or a business, depends on how you are related to it. The level of abstraction you apply when making decisions, influences the outcome. Are you able to see the bigger picture, or are you afflicted by tunnel vision while solving a specific task, without caring about your surroundings? As a programmer in a development team, your primary task is to design and implement the code required to solve the various tasks. As a SCRUM Master, your job is to ensure that the team follows the right processes and are working on the right things in accordance with the customers wishes. As the company owner, it is your responsibility that projects are executed according to the rules of the business they are executed in. The programmer, SCRUM Master and product owner represents the three abstraction levels. The programmer represents the product, the SCRUM master represents the product's business model (the project) and the product owner represents the business pattern (the business). While the product and project level were discussed in the first [11] and second [13] reports, this chapter will describe what business patterns are, what good they do, why they are needed, and how they are applied to not only the Cloud Apps project, but Meebox as its owner as a whole. In order to understand the full impact of a change in the vision, you must understand the business it is placed in. The business patterns services that purpose. This chapter explains what the business patterns are and how they should be used, and Chapter 7 explains how they are integrated into the Vision Development methodology.

Osterwalder, A. and Pigneur, Y. presented five different business patterns in the Business Model Generation [24] that were used in the first report to make a business model for both Cloud Apps and Meebox. Those patterns are the unbundled business pattern, the long tail, multi-sided, freemium and open business model. Cloud Apps part 1 [11] described the Cloud Apps product as being based on a multi-sided business model, because there were three dependent customers; the hosting providers, the application developers and the end customer. At the time of the first report, the task was to identify the Cloud Apps business model how that fits into Meebox' business. Who are the customers? What are the products value propositions? What are the costs? Why should the customer choose this product?

If we look at the whole business instead of the individual parts that it is composed of, the five different patterns from the Business Model

Generation are not sufficient to cover the different business patterns you can use. There is no right or wrong business pattern, however. It depends on how the owners/managers chooses to structure the business.

Gassmann, Frankenberger and Csik are discussing the St. Gallen Business Model Navigator in their 2014 article [15]. The claim is that the future business battlefield will consists more of business models and business patterns than of technological advantages or providing a service at the best price, as it has previously been seen. Companies that are stuck in the past or are caught off guard by changing markets, technologies or competitors will loose in the long term. An example of this is *Kodak*. When their first digital camera was released in 1999, they forecasted that ten years later, digital cameras would only account for 5% of the market. The reality turned out to be just the opposite, as in 2009 95% of all cameras were digital cameras and only 5% were analog cameras - the ones that Kodak put their business in. This meant that in the same period, Kodak fired 80% of all their employees before finally filing for bankruptcy in 2012 because they were not prepared to change their business and follow the trends. Learning from past mistakes, a survey by IBM in 2012 showed that over 90% of the participating CEO's are either considering or planning to innovate their business models withing the next three years [16]. One thing is to get on top, but it is a different story to stay on top. Staying on top requires the ability and courage to rethink, redesign and adapt a business to future needs.

Gassmann, Frankenberger and Csik describes 55 different business patterns, the different companies that has implemented them and what part of a business they describe. To describe this, they are using a four-dimension model.

4.1 WHO, WHAT, HOW & VALUE

The model is reduced from Osterwalder and Pigneur's nine dimensions to four dimensions, to make it easier to use but, at the same time, make sure it is exhaustive enough to give a clear picture of the different patterns.

- The "Who" describes who the customers are. It is important to know who the customers are in order to target them correctly. If we do not know who the customers are, we do not have a chance to create something they will want to buy.
- The "What" describes what is offered to the customer, the products value propositions. It is the company's bundle of products and services that are of value to the customer.

- The "How" describes the different processes and activities that the business has to master in order to deliver the value propositions to the customer.
- The "Value" describes how the business makes money and why it is financially viable, relating to the revenue model. It uses cost structures and applied revenue mechanisms to display how the business makes money.

The purpose of answering those four questions is to have the required knowledge about the products to create its current business model. Only when the current model is known can it be subject to re-thinking and innovation.

Meebox' "who, what, how and value" was answered in Cloud Apps part 1 [11] using the business model canvas from Osterwalder and Pigneur [24]. The "who, what, how and value" aims to describe the same as the Business Model Canvas, however I believe that the Business Model Canvas can give a better and deeper description due to it having nine descriptive areas instead of four. Translated from the business model canvas into the four dimensions, Meebox' are:

- The "Who". The customers segment is defined as:
  - End Customers. They are targeted using SEO + online advertisement, social media activity, purchases through the Cloud Apps website and post-sales activities. They are identified by:
    - Mass Market – due to the generic applications available in Cloud Apps.
    - Niche Market – due to the ability to publish limited applications.
    - Small/medium Business and entrepreneurs.
  - Application developers. They are targeted through developer forums. They are identified by:
    - Niche Market – targets application developers who wants to public their web-applications on the platform.
    - Entrepreneurs and small/medium businesses.
  - Hosting providers. They are targeted through personal sales with account managers. They are identified by:
    - Niche Market – targets Cloud Hosting providers who can allow end customers to provision new Cloud Apps on their platform.
- The "What". The product has different value proportions to the different customer segments defined above. The "What" is:

  - End Customer: Easy application deployment, data integrity, good performance, automatic updates and scaling, usability and cheap prices.
  - Application Developers: Newness - the Cloud Apps marketplace gives them a new platform to offer their application to the end customers on.
  - Hosting Providers: Easy access to new markets. The Hosting Providers have to keep their cloud environments running, and then the Cloud Apps system will provide them with new customers.

- The "How". The project has to master a number of task, activities, resources and partners. They are identified as:
  - Platform development and maintenance. The systems running the Cloud Apps platform has to be maintained and kept up to date.
  - Marketing. The system will have to be marketed to reach its target audience.
  - Key partners in application developers and hosting providers (who are also customers) should be nursed and taken extra care of, as they are important parts of the business model.

- The "Value". The following bullets describes how the product makes money.
  - End Customers: Fixed pricing and subscription fees. The end user pays a fixed fee e.g. every month or in another interval to keep his service running. As long as a end customer has an active Cloud App instance, it is recurring income.
  - Application Developers: There is no direct value gained from this segment. They generate value by providing applications that the end customers will purchase and through that provide value.
  - Hosting Providers: A fixed one-time setup fee. No running or renewing fees, but indirect value from the end customers provisioning new Cloud App instances on their platforms.

When we know what each of the four dimensions contains, we can identify which patterns applies to the business model.

4.2 55 PATTERNS

Gassmann, Frankenberger and Csik presents 55 different business patterns in the article, St. Gallen Business Model Navigator [15]. They

are derived from analyzing a number of international companies and how they have each evolved during their lifetime, transforming between the different patterns that describes how the business works. It does not make sense to list all 55 patterns here, so I will only list the ones that applies to this project, the Cloud Apps, and discuss why they match. With the knowledge of what business patterns applies to the project, it is possible to compare it to the patterns that fits Meebox (they will be discussed in Chapter 8). It will also be possible to investigate how the Cloud Apps product itself has evolved, and what direction it may have pulled the entire company in.

The patterns in the following list are the ones we have found to match this project. The text snippets in quotes are the original descriptions from the business pattern navigation.

- **Pattern 6: Cash Machine** – Relates to How and Value. "In the Cash Machine concept, the customer pays upfront for the products sold to the customer before the company is able to cover the associated expenses. This results in increased liquidity which can be used to amortise debt or to fund investments in other areas." [15].
  The customer pays up-front for a period of their choice (e.g. 1, 3, 6 or 12 months), however the product is delivered as a service over time.

- **Pattern 20: Guaranteed Availability** – Relates to What, How and Value. "Within this model, the availability of a product or service is guaranteed, resulting in almost zero downtime. The customer can use the offering as required, which minimizes losses resulting from downtime. The company uses expertise and economies of scale to lower operation costs and achieve these availability levels." [15].
  Cloud Apps offers an application as a service that the customer bases important infrastructure on, so it has to be available all the time. Availability and uptime are key value propositions.

- **Pattern 34: Orchestrator** – Relates to How and Value. "Within this model, the company's focus is on the core competencies in the value chain. The other value chain segments are outsourced and actively coordinated. This allows the company to reduce costs and benefit from the suppliers' economies of scale. Furthermore, the focus on core competencies can increase performance." [15].
  Two of the important parts of the product, the hosting and application, are outsourced. Hosting Providers offers their hosting infrastructure to host the Cloud App instances and the Application Developers create and offer applications that can be installed.

- **Pattern 35: Pay per Use** – Relates to What, How and Value. "In this model, the actual usage of a service or product is metered. The customer pays on the basis of what he or she effectively consumes. The company is able to attract customers who wish to benefit from the additional flexibility, which might be priced higher." [15].
  A key value proposition of the Cloud Apps is its ability to detect the required resources to deliver an acceptable performance. The customer pays the price for the allocated resources, and because they are allocated and removed based on what is required, it is pay per use.

- **Pattern 49: Super Market** – Relates to What and Value. "A company sells a large variety of readily available products and accessories under one roof. Generally, the assortment of products is large but the prices are kept low. More customers are attracted due to the great range on offer, while economies of scope yield advantages for the company." [15].
  The Cloud Apps marketplace will offer a wide variety of applications that the customer can choose to install on the Cloud Apps instance.

- **Pattern 52: Two Sided Market** – Relates to What, How and Value. "A two-sided market facilitates interactions between multiple interdependent groups of customers. The value of the platform increases as more groups or as more individual members of each group are using it. The two sides usually come from disparate groups, e.g., businesses and private interest groups." [15].
  The pattern is called "two sided market", but this is actually a three sided market that involves the End Customer, purchasing the Cloud Apps Instances, the Hosting Providers, offering their hosting infrastructure and the Application Developers, creating applications for the Cloud Apps Instances.

Knowing what patterns apply to the product enables us to determine whether this fits into the business. If the analysis is repeated frequently, managers should also be able to detect any changes in the patterns that applies to the business and react on the changes. Changes in what patterns applies to a business, means that the business has evolved. Only the managers/owners can decide in the given situation whether that is a good thing or not, but having the knowledge and knowing what the original vision for both the business and individual project was, gives them the information they need to decide on it. That is why we need the business patterns: To identify the current state of the product, if it fits into the customers business, how the product and business are evolving and if the evolving is going in an acceptable direction.

The next chapter will discuss different types of criteria and tools to determine whether something is wanted or not. Specifically it will present tools to help identify if a new idea is desirable or not. Those tools will be used in the vision development.

5 IDEA EVALUATION

An idea is matured over time by constantly revising it, evaluating if it is still a good idea, and remove the parts of it that are not. Such evaluations are made using criteria and tools for evaluation. Evaluation of an idea is closely related to evaluation of a vision, in that the concept of creating a vision in this report is largely based on using new ideas. This chapter will discuss different tools and methods used to evaluate ideas, but also discuss the criteria used to determine if something is a good idea or not. That will be used in Chapter 7 when new ideas are suggested and it should be determined whether or not to accept them.

There is no fixed list of criteria that can be checked off to see if an idea is good or not. The list of criteria varies from project to project, from business to business, depending on the individual *vision* of that particular product, project or business.

This chapter will start by presenting a number of idea evaluation tools and methods. They are originally presented among 32 different tools and methods for idea evaluation in the "Idea evaluation methods and techniques" paper by dr. Miroslav Rebernik and Barbara Bradac [27]. Out of the 32 tools and methods, the most relevant for the Vision Development methodology will be presented below and described how they can be used. After that, this chapter describes the Vision Pattern, a tool that can be used to understand a vision during maturation. This is also used in Chapter 7 to better understand what it is that the vision describes. Finally, composite ideas are discussed, as the it is closely related to a vision. Therefore, tools used to evaluate a composite idea can also be used to evaluate a vision, which is shown in Section 5.4.

5.1 CHECKLISTS FOR IDEA EVALUATION

One common way to evaluate ideas and business opportunities, is to create a checklist of questions to be answered in relation to the idea [27]. They are meant to check if the idea is eligible, feasible and economically well-founded. Below are two checklists designed to evaluate an idea for a business or product and for evaluating a new product idea.

Evaluating an idea for a business or product

Princeton Creative Research[1] has developed a criteria checklist for evaluating ideas for a business or product. The purpose of asking these questions is to identify whether an idea is plausible to work with. The original checklist can be seen in [27] on page 25. Table 1 shows the checklist. The first fourteen questions are the original questions form the checklist, while no. fifteen has been added to account for the impact that an idea can have on the product and/or business model.

The purpose of the questions in Table 1 is to evaluate the idea/vision change from a business-point-of-view. The questions addresses general aspects of the idea and tests if it can succeed both in the business and on the market. The appropriate place and time for these questions are discussed in Chapter 7.

Evaluating new product idea

The 21-point invention evaluation checklist[2] will help to identify the chances of success with a new product or idea. It can be adapted to evaluating a vision change by removing some questions, modifying others and finally adding some too. The checklist can be seen in Table 2.

Table 2 has 19 questions that should be asked in relation to a new idea. While the original checklist consisted of 21 questions, three has been removed, one added and multiple has been modified from focusing on a specific product to focusing on an idea that will affect a product and/or vision. The modifications are mentioned in the footnotes. Two irrelevant questions have been removed. The questions that are not mentioned are original from the list. The words have been modified to be more relevant to software development and product ideas, but the essence of the questions are the same. The purpose of asking these 19 questions is to get a broader view on the idea from both general, industrial, market and product specific criteria.

The appropriate place and time for these questions are discussed in Chapter 7.

1 http://www.entrepreneur.com/encyclopedia/checklists/article81940.html
2 http://www.entrepreneur.com/encyclopedia/checklists/article81922.html
3 Modified to focus on possible value additions by the product.
4 Modified to focus on possible value additions by the product.
5 Modified to ask how the product affects the industry.
6 Modified to ask how the product affects the industry.
7 Modified to ask how the product affects the industry.
8 Modified to focus on what impact the product has on the market.
9 Modified to focus on what impact the product has on the market.
10 Modified to focus on what impact the product has on the market.
11 Modified to focus on what impact the product has on the market.

#	Criteria Question	Answers Arguments
1	Have you considered all the advantages or benefits of the idea? Is there a real need for it?	
2	Have you pinpointed the exact problems or difficulties your idea is expected to solve?	
3	Is your idea an original, new concept, or is it a new combination or adaptation?	
4	What immediate or short-range gains or results can be anticipated? Are the projected returns adequate? Are the risk factors acceptable?	
5	What long-range benefits can be anticipated?	
6	Have you checked the idea for faults or limitations?	
7	Are there any problems the idea might create? What are the changes involved?	
8	How simple or complex is going to be the idea's execution or implementation?	
9	Could you work out several variations of the idea? Could you offer alternative ideas?	
10	Does your idea have a natural sales appeal? Is the market ready for it? Can customers afford it? Will they buy it? Is there a timing factor?	
11	What, if anything, is your competition doing in this area? Can your company be competitive?	
12	Have you considered the possibility of user resistance or difficulties?	
13	Does your idea fill a real need, or does the need have to be created through promotional and advertising efforts?	
14	How soon could the idea be put into operation?	
15	What impact will your idea have on the product/business?	

Table 1.: Checklist for Idea Evaluation

5.2 SWOT

A SWOT analysis is a method to evaluate the strengths, weaknesses, opportunities and threats related to an object. An object can in this context be anything that needs to be evaluated, e.g. a product ("how

#	Criteria Question	Answers Arguments
	General Criteria	
1	Is your idea legal?	
2	What is its environmental impact?	
3	Will it improve product safety?[3]	
4	Will it improve product quality?[4]	
5	Will it have wide social acceptance?	
6	Will it have any negative impact?	
	Industry Criteria	
7	Will it add or remove competitors?[5]	
8	Will it add or eliminate required assistance from other products?[6]	
9	Will it affect the pricing?[7]	
	Market Criteria	
10	Does the idea fit into a trend?	
11	Will the idea fulfill a need or create new needs?[8]	
12	Will it add/remove seasonal effects?[9]	
13	Will it add or remove potential customers?[10]	
14	Will it add or remove the need for instructions?[11]	
	Product Criteria	
15	Will it increase or decrease the cost to get the product to market?	
16	Will it add or remove requirements for service and maintenance?	
17	Is there a warranty?	
18	Does it need packaging?	
19	Will it make the product more simple or more complex?	

Table 2.: Checklist for new Product Idea

will this new product do on the market?"), a person ("should we hire a new person?") or business ("simplified core values of a business, how does a given business fit into an industry?"). It takes both internal factors (strengths and weaknesses) and external factors (opportunities and threats) into account during evaluation.

A SWOT model is based on four squares capturing the strengths, weaknesses, opportunities and threats, and in doing so also categorizing them in internal and external and helpful and harmful. An example of a SWOT model is shown in Figure 2.

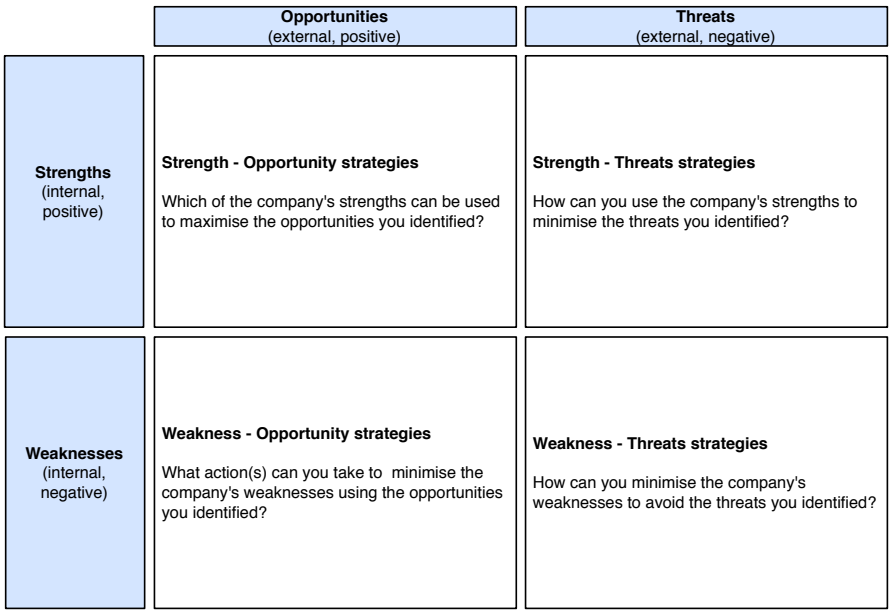


Figure 2.: SWOT analysis model.

The four squares has a strategical fit to cover all aspects of the object being evaluated. They each cover:

- Strengths: Describes what gives the object an advantage over others.
- Weaknesses: Describes how the object is weaker than others or what gives it a relative disadvantage compared to others.
- Opportunities: Different elements outside of the object in surroundings that could be exploited to its advantage.
- Threats: Elements in the environment that could show to be a threat and cause trouble for the object.

In the article "How to perform a SWOT analysis" by Tim Berry, CEO of Palo Alto Software [8], he suggests a number of questions that can be asked in every box of the SWOT analysis. The ones that are relevant are shown below and should be used when using a SWOT analysis on an idea in vision development. The questions will be based on evaluation of a product vision, but can be modified to fit any object.

- Strengths (internal, positive)

  - How is this product stronger than the competitive products?
  - What current and future advantages does this product have?
  - How can this product strengthen the business?
    * Automate internal processes (reduce costs).
    * Boost sales.
    * Raise employee happiness?
  - What facilities does the business have to support this new product?
    * Good development facilities.
    * Money to make it better, faster, cheaper to sell e.g.
  - What other positive factors will the product add to the business?

- Weaknesses (internal, negative)
  - What factors within the business prevents us from creating this product?
    * Lack of money to support development.
    * Unqualified employees.
    * Lack of resources or internal support.
  - What factors prevents this product from gaining a competitive edge on the market?
  - What other negative factors will affect the product?

- Opportunities (external, positive)
  - What opportunities exists in the current environment or market that we can benefit from?
  - How will the market react to the new product?
    * If positive, how can we benefit from that?
    * If negative, how can we change that to positive?
  - Is the opportunities ongoing or do they only exists in a small window of time?

- Threats (external, negative)
  - What competitive products exists in the market today?
  - What factors beyond our control can place the business at risk?
  - What external situations could threat the products place on the market?
    * Changes in economy.

    * Changes in the government.
    * Changes in laws and regulations.
    * Changes in customers needs and behavior.
  - Has there been significant changes in suppliers prices, quality or their deliveries or other factors related to them?
  - Has newly introduced products or technologies made our product obsolete?

Those questions should spawn discussions that throws light on the product vision as it currently is and help the team evaluate it with the purpose of improving it.

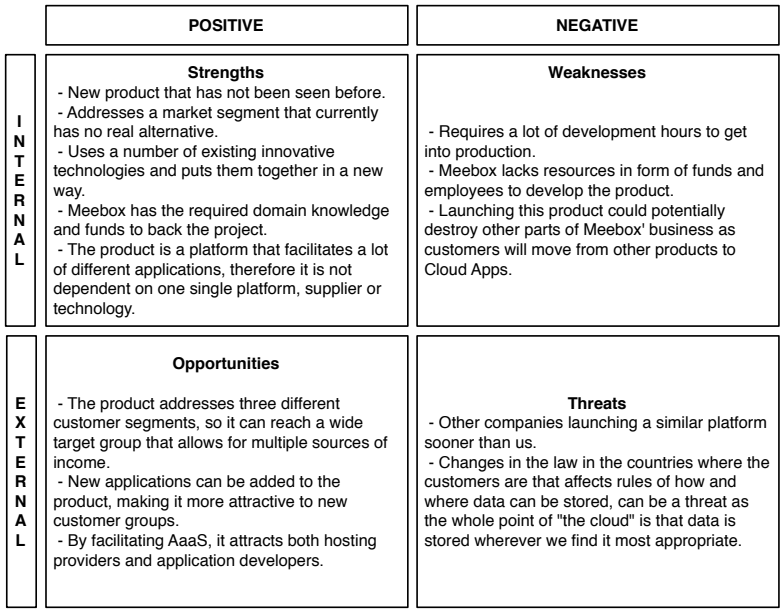


Figure 3.: "SWOT analysis" of the Cloud Apps vision

Figure 3 shows a SWOT analysis of the current Cloud Apps vision. This should be used to strengthen the product. The strengths should be build upon. The weaknesses eliminated. The opportunities hunted to be realized. The threats should be shielded against.

When the product and vision develops, a new SWOT analysis should be created to identify how they have changed and what we now should be aware of. Using this tool as described in Chapter 7 will help to evaluate the current vision. By knowing the state of it, it can be evaluated if the current state is a desired development from the previous state, and depending on that it will affect how we work with the product and vision in the future.

5.3 THE VISION PATTERN

The Vision Pattern originates from Essence [2] as a tool for vision maturation. It is used to answer a number of questions that arises when working with a project vision, such as: How do we know if this vision makes sense? How do we assess its qualities and limitations? - and present them in a logical and easily accessible way. The Vision Pattern provides a way to see the Vision in its context and determine if it makes sense as a part of its maturation.

The Vision Pattern is a visual expression of the main ideas behind the project, capturing the main ideas into a model that reasons about both inner and outer aspects. The Vision Pattern is designed as an argumentation model that is intended to give a structured representation of the Vision. The Vision Pattern is based on the Toulmin Model of Argumentation, originally developed by the British philosopher Stephen Toulmin. In his model, Toulmin identified a set of interrelated elements of persuasive arguments for analyzing complex rhetorical arguments. Those arguments makes the Vision Pattern capable of both producing arguments as well as provide convincing justification for a claim.

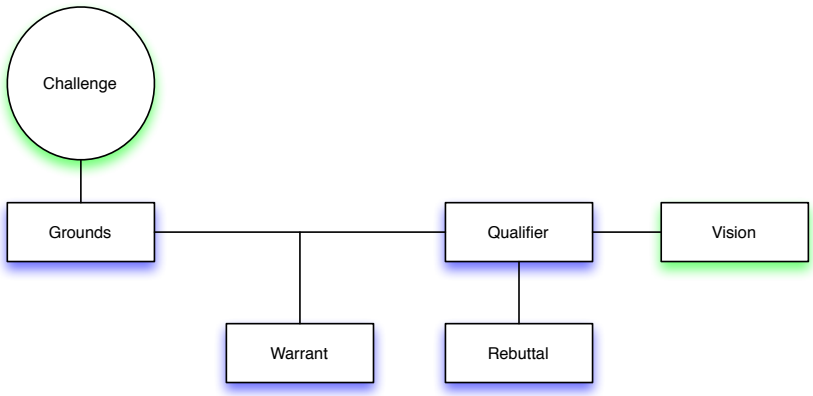


Figure 4.: The Vision Pattern template from Essence

The template for the Vision Pattern can be seen in Figure 4. The rectangular objects in the model are the original elements from the Toulmin Model of Argumentation. The circular element is added by Essence and contains the Challenge that we have set out to answer (the problem that the customer wants solved). The Challenge and Vision elements are both green to illustrate that they are closely related. The Vision is a direct answer to the Challenge, in that it describes the product that solves the problem the customer presents. The blue elements are the argumentation behind the model. The Vision Pattern consists of six elements that will each be explained below.

- **Challenge**. This element originates from Essence and describes the overall problem to be addressed. It is not a part of the original argumentation model, but is added to define the context of the Vision. While the Challenge defines the context of the vision, it does not define the scope. Therefore the Vision can potentially cover more than the Challenge does, thus provide more value and benefits than just solving the Challenge. Although it is possible, it is more likely that the Challenge often is too comprehensive and the Vision only offers a small partial response to it.

- **Grounds**. The Grounds represent the basic needs that are related to the Challenge and should be fulfilled by the project. It reasons about- or provides supporting evidence that bolsters the Vision. The Grounds describes the reality that the Vision is based upon.

- **Warrant** The Warrants describes why it is important to fulfill the needs described in the Challenge. It argues the relevance of the Grounds in relation to the Vision and thereby legitimizes it. It describes why the evidence in the Grounds supports the Vision and points out why exactly it is both relevant and important.

- **Qualifier**. The Qualifier is best described as the arguments that are assumed present or required to qualify the Vision. E.g. if the Vision says that we want to send an e-mail, then the Qualifier could say "if we have Internet-connection".

- **Rebuttal**. The Rebuttal is explaining why the qualifier is valid and thus not an issue. It is therefore closely linked to the Qualifier. This is also shown in the model, where the Rebuttal element is placed just below the Qualifier. Using the same example in the Qualifier, the Rebuttal could say "it is no problem as we will probably always have Internet-connection".

- **Vision**. Toulmin's original Model of Argumentation contains an element named "Claim". That has been replaced with Vision in this model, as it is the Vision, or the change to the vision, we are trying to reason about. The Vision contains a either whole or partial proposed response to the Challenge. This can be either the overall challenge, or it can be a new idea being evaluated, in which case the Vision in this model only represents the change. The Vision is dynamic by nature and may change over time when new knowledge, insight or outside factors reflects on the project.

The Vision Pattern's purpose is to always represent the reason of the project as it is currently understood. It can (and should) mature over time in the same pace as the Vision does and function as an

assisting tool of maturation to explore other sides of the Vision and reason about it. All parties involved in the project can participate in the discussions about the Vision Pattern, each contributing with the knowledge from their respective fields. Doing so should help them see the project from new angles and ask questions that they may not have asked otherwise, and hereby considering the project from different perspectives.

Using the Vision Pattern as described in Chapter 7 will help the team to reason about the current Vision compared to the product that they are actually working with, and it will help them to mature and adapt the Vision to new input, requirements or other changes that will affect the project.

5.4 COMPOSITE IDEA EVALUATION

Sections 5.1, 5.2 and 5.3 all discusses evaluation of a single idea. However, a product vision is not a single idea. A product vision is can be the result of a small number of central ideas and a larger number of other ideas that are bundled together into a vision. Take the product vision for the Cloud Apps project as an example:

> *" A Cloud App is an Application that is hosted in the Cloud in a self-contained isolated environment. It is automatically provisioned upon request and it automatically scales its resources to handle the load at any given time. Furthermore, it is automatically updated and maintained. "*
>
> — Definition of Cloud Apps, Cloud Apps Report 1 p. 8

Breaking that vision down, it shows a number of ideas that has been brought together to form the vision:

- An Application hosted in the Cloud in an isolated container. That has been available for some time, and it is easy to manually install an application on a cloud server.
- Automatic provisioning and automatic scaling. Some systems are able to automatic provision virtual servers and applications. Some systems are also able to automatically scale. However, the majority of these are hardcoded to work with a given web application.
- Automatic updates and maintenance. Few systems can flawlessly handle automatic updates and maintenance. Often, web applications are composed of a core system and a number of plugins or addons developed by third parties. This makes automatic updates more difficult, as there are multiple unknown factors to take into account.

Add to that the idea of adding multiple hosting providers and the Cloud Apps Marketplace with a number of open source applications available, plus application developers who can develop applications specifically to the Cloud Apps Marketplace. Then we have a number of individual ideas that, when brought together, forms the product vision for the product.

Because a vision is composed of a number of individual ideas, and these individual ideas often are the result of an elimination process as described in the previous sections, where multiple ideas that answers the same question are compared and one is finally chosen, it is important to know how to compare ideas.

A NAF-evaluation *[Novelty, Attractiveness and Feasibility]* scores different ideas based on their novelty, attractiveness and feasibility. Each score is between 1 and 5 where 1 is lowest and 5 is highest. The total score of each idea will give its ranking compared to the others. Table 3 shows an example of a NAF-evaluation that considers how to ensure performance on a Cloud Apps instance.

Idea	Novelty	Attractiveness	Feasibility	Score
More powerful servers	1	4	2	8
Load balancing across multiple smaller servers	3	4	2	9
Automatic scaling of resources	4	5	4	13
Limiting the number of simultaneous connections	1	1	4	6

Table 3.: Evaluation of ideas to ensure performance on a Cloud Apps instance using a NAF-evaluation

The first idea of more powerful servers is a classic solution to ensuring more performance. To ensure that the application always has enough resources the developers estimates the load that it will generate and scales the server running the application somewhat above that expected load. However, it has been seen many times before and it is a very expensive solution, because the customer always pays for the big server, no matter if he uses its resources or not. On the other hand, it is a quite attractive solution as it ensures performance. Therefore it gets a total of 8.

The second idea of load balancing the application across multiple smaller server has also been seen before, but not as often because it

can be complicated to implement. It is a very attractive idea because it is very flexible, but it is not necessarily a very feasible one because it can be very difficult to configure and thus have a high start-up cost. Also, this idea requires technicians to set up new servers or shut existing ones down when the load increases/decreases. Therefore the total score is 9.

The third idea of automatic scaling the resources allocated to the server running the application is quite new, and therefore gets a 4 in novelty. It is a very attractive idea because it gives the customers flexibility in both performance and pricing, as the system will ensure that it always has the correct amount of resources and is billed accordingly. Therefore this idea gets a total score of 13.

The final idea is limiting the number of simultaneous connections. The idea scores very low in both novelty and attractiveness because it has often been seen before and because it is not very attractive to tell the customers: "We can only handle X simultaneous visitors, you are Y in line". If that happens, visitors will disappear quickly. Finally it has a 4 in feasibility because it will be a very cheap solution to implement and maintain, but it may have a big impact on the economics in the application. Therefore this idea get a total score of 6.

The idea of automatically scaling the resources allocated to a Cloud Apps instance wins. The next step is to dig into this idea and mature it. The first step in that is to go back to the Paradigm- and Product Views from Essence, and use those to generate ideas and examples on how to implement this idea. While Essence suggests using either the NAF-evaluation or a more general PMI *[Plus, Minus and Interesting]*, I prefer a slightly modified version of the SWOT-analysis 5.2. This version of SWOT scores each of the four fields in the following way:

- Strengths are scored 1 to 5, where 5 is a very important strength and 1 is a minor or irrelevant strength.
- Weaknesses are ranked 1 to 5, where 1 is a big weakness and 5 is a minor or irrelevant weakness.
- Opportunities are ranked 1 to 5, where 5 is an important opportunity and 1 is a minor or irrelevant opportunity.
- Threats are ranked 1 to 5, where 1 is a big threat and 5 is a minor or irrelevant threat.

By applying such a SWOT analysis to each of the suggested answers to the challenge (the idea that was brought to the Paradigm and Product View of Essence), it is possible to compare them and choose the best one at the time of the evaluation. Table 4 contains an example of some of the answers to the "Automatic scaling of resources" challenge that was chosen during the development of the Cloud Apps product.

Idea	S	W	O	T	Total
Purchasing third party software	5	2	4	5	15
Creating the scaling algorithm internally	4	3	4	5	16
Hiring consultants to create the scaling algorithm for us	5	2	3	3	13

Table 4.: Evaluation of strengths, weaknesses, opportunities and threats to Automatic scaling of resources, using a point-based SWOT analysis for multiple idea evaluation and comparison

Table 4 shows that it would be the best idea to create the algorithm for scaling the resources of a Cloud Apps instance internally if possible. It has the strength of a lot of flexibility and that we can create it to work exactly how we want. The weaknesses are few, but the biggest one is that it will require a lot of resources to develop and maintain. There are lots of opportunities both short- and long term to either develop this algorithm internally or perhaps commercialize it later on. Finally the threats are few too, as there will be few competitors and we can modify the algorithm along the way as we know and understand how it has been built and is working.

Using these tools to take single ideas, expand them to composite ideas, evaluate each of them and finally putting them together into a vision, enables us to ensure that the vision is composed of the right elements. This only shows a fraction of the evaluation work done on the Cloud Apps vision, to give examples. It will be elaborated upon later with concrete examples and descriptions in Chapter 7.

This chapter has described three concrete tools and methods for evaluating- and working with an idea and/or vision. The tools have merely been described here, explaining how they work in the context of a software development project. The tools will be used in Chapter 7 where it will be described how they are used before, during and after the development of a project.

The purpose of this first Part of the report was to gather and explain the knowledge required to understand and work with vision development during a software development project. How the tools and methods discussed in this part were used in practice is described in the next Part of the report.

Part II

METHODOLOGY

This part of the project will use the Cloud Apps project as a case study, for finding a method to determine if a vision has evolved, and if it has, what to do about it. The goal is not only to analyze the Cloud Apps project, but also contribute with a new methodology that can be applied on other projects, to determine if the vision that is guiding the project is the right one, and if it is not, what can or should be done about it.

The previous chapters provided the required knowledge about software development, business patterns and how to define a vision in relation to software development projects. That knowledge will be used in the following chapters as foundation for the discussion of vision development.

This part consists of three chapters. The first, Abstractions, will talk about the different abstraction levels that we will be working on when working with the Vision Development methodology. The next is the Vision Development chapters that describes the theory behind the methodology, what it contains and how to apply it. Finally the third chapter describes Cloud Apps as a case study where the methodology has been applied.

6 ABSTRACTIONS

The methodology presented in this part of the report, works with a product vision from three different types of abstractions: The *product*, focusing on design- and implementation details of the product. The *project*, focusing on both how the product fits into the business, who its customers are, its value propositions etc. using the business model, but also the business as a whole. Finally the *business* that focuses on the business as a whole using business patterns and how the product and products business model is either affected by the business patterns or how they themselves affect the business patterns.

Because we will be working with these three different types of abstractions, and because we need to adapt different views on the project depending on what we are working with, the three abstractions are each described in the following sections to clarify where they apply and how to work with the vision with each of them.

6.1 BUSINESS

Chapter 4 described what a business pattern is and why a business should identify which patterns applies to it. The Who, What, How and Value (see Section 4.1) describes the purpose of the business from an overall perspective and is used to give an idea about the kind of business that we are dealing with. The 55 business patterns that are derived from the St. Gallen Business Model Navigator [15] were discussed and the ones that applies to the business that could develop, maintain and sell Cloud Apps were presented and discussed.

Decisions made on any of the three types of abstraction will affect the other two abstractions. Decisions made on the business level can have wide reaching consequences, affecting both products, clients and employees. Therefore often only a limited group of people are able to make decisions directly affecting which patterns will apply to the business, and the developers, engineers and designers working on a concrete project or product are usually not a part of that group.

The reason why this abstraction is important to incorporate into the concrete product vision is that, while you as a developer may not be able to affect the business patterns, you have to work in accordance with them because they define the customers business.

The business patterns have not been discussed in the first two Cloud Apps reports. While the first focused on the project and business model and the second on the product, this report takes a broader approach to the development of Cloud Apps from both a development- and business point of view.

The people working with the business patterns are typically managers, owners and to some extent product owners. It will be used during project initiation and project review, where the product/project is evaluated against the original targets.

6.2 PROJECT

A lot of focus was put on the project and its business model in the first report on Cloud Apps [11] where both business models for the product and business was presented. The business model for the product answers questions like: What is this product? How does it fit into the business? What are its value propositions, potential customers, pros, cons etc. This was briefly recapped in Chapter 1.2 that described the business model and the Cloud Apps idea from the first two reports.

The Cloud Apps project has been implementing its business model based on the Business Model Canvas by Osterwalder and Pigneur [24].

The people who will be working with the business model are typically managers, product owners and, if it is a product specific business model, the customer. Their input helps design the initial business model. The business model is dynamic in the same way as the vision. Many internal as well as external factors can affect it, such as new technologies, ideas, suggestions, new competitors, potential clients or even elements out of our immediate control such as national economies etc. Therefore the business model must be dynamic in order for it to be transformed if, or when, needed. Typically the business model is used during the project initiation, sprint planning, sprint review and project review.

6.3 PRODUCT

The final abstraction is the third and most concrete one: The product. While the Business Pattern describes the business as a whole, and the Business Model works with the Business Model Canvas to describe the business model for a company, project or product, the Product abstraction works with the product vision and the product itself, including the design, implementation and testing etc.

When we are working on the product, we have to be prepared to implement changes from the outside, but it is also on this abstraction that new ideas, technologies or ways to do things are discovered and

can be implemented.

The people who will be working with the product on this level are the product owner, SCRUM Master and developers. It focuses on the SCRUM work flow, project initiation, iteration planning, execution and evaluation and day-to-day work and decisions.

The purpose of this chapter was to describe the three different abstractions that we will be working with in the methodology and how the abstractions relate to each other. The next chapters will describe how the methodology works and presents as a case study on the Cloud Apps project.

7

VISION DEVELOPMENT

Everyone working on a project should be able to ask the right questions at the right time to understand the impact of their decisions.Doing so should enable the team to give intelligent answers to important and complex questions. In this context, an intelligent answer is considered an answer, where the one answering the question is aware of both pros and cons of each possible answer to the question. By answering on an informed basis, it is less likely that one will give wrong answers to questions, which can have negative consequences beyond immediate scope of the question. Answering some of the aforementioned questions will help reveal if the product is moving in a different direction than the vision originally intended. There can be different reasons to why the product is moving in a different direction.

- Inner changes, e.g. engineers are being innovative and are discovering new technologies or possible solutions to existing challenges that moves the product in a new direction.
- Outer changes, e.g. the product owner that brings new requirements or feature requests to the product as a consequence of changing markets or new requirements from the customers.

When it is made clear whether the product is moving in another direction than the vision originally stated, one is able to ask whether the diversion from the current vision is desirable or not. Such diversions are what *vision development* helps to identify and handle. If changes to the vision are desirable, the vision should be modified to embrace the changes and the new direction. If changes to the vision are not desirable, the change should be discarded from both the vision and product. If either the product, business model or business patterns are being modified, the modification on one of them can affect the other two. That is why it is important to understand the impact of the decisions you make, made on either abstraction, because it can affect parts of the business that are not directly linked to the abstraction you is working with.

This chapter is the theoretical foundation of the methodology behind vision development that I have used in the Cloud Apps project. Based on the knowledge that was gained in the previous chapters, this chapter will describe how the methodology is applied in the different phases of a project, and what should be in focus at what time. The methodology will separate the different abstractions, and give a clear overview of when to discuss each one of them.

The methodology is based on SCRUM (see Section 2.2). On top of SCRUM is Essence (see Section 2.3) to facilitate software innovation. SCRUM's process of handling the development and Essence's roles and views are the foundation on top of which the Vision Development methodology is built. Vision Development adds tools and methods to identify vision changes and how to react to the changes.

Figure 5 shows the vision development model. The figure is split into four sections:

- **Project Initiation**. During the Project Initiation it is determined what we will be working with in this project. What is the product, what are our expectations and requirements and what do we expect to deliver upon completion.
- **Sprint Planning**. This is an iterative processes that occurs once every sprint as the first activity. The items from the backlog are included into the sprint backlog and tasks are assigned to team members. Sprint goals are set.
- **Sprint Execution & Review**. This part contains two phases, namely the sprint execution and sprint review. What should we execute "today" and during the whole sprint? Did we reach the sprint goal?
- **Project Review**. When all sprints are completed and the product delivered, the project review looks at what has been delivered and if it is in line with what we originally expected from the project.

The four phases are discussed separately in the following sections.

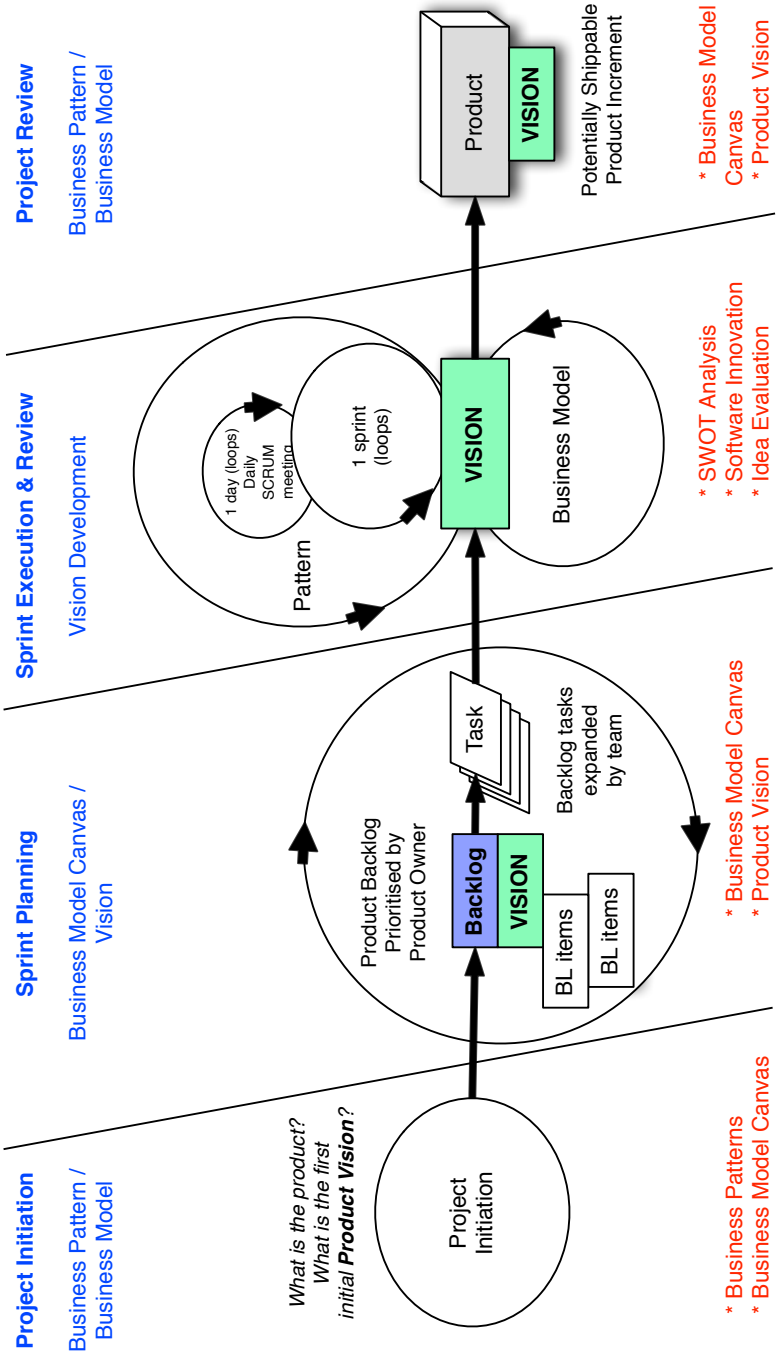


Figure 5.: Vision Development Model

7.1 PROJECT INITIATION

This section will describe the first phase in Vision Development. This phase is linked to the initial steps that one does before starting a project and a SCRUM process. The phase can be seen in Figure 6, which is a part of the Vision Development Model in Figure 5.

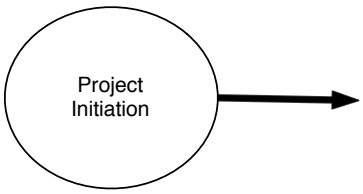


Figure 6.: Vision Development, Phase 1: Project Initiation

When it has been decided to start a project, the first step in the project is the Project Initiation. The Project Initiations first step is to clarify exactly what the expected output of the project is. The Product Owner and SCRUM Master should ask themselves two questions before the project is started:

- What is the expected output of this project?
- Does the expected output fit into the business model and the business patterns?

The first question is to some extent a negotiation between the Product Owner and the SCRUM master. What is realistic to deliver within the scope of the project, and how does that compare to the Product Owners expectations? The purpose of the first question is to align expectations from the start, and have a common ground from which the project can be kicked off. The purpose of the second question is to figure out what the product is, what both parties think it is and whether it fits into the business as it is today. To answer this it is necessary to include the Business Model and Business Patterns and compare them to the expected output. This will answer a number of questions related to whether the expected output fits into the existing business model and business patterns:

- Does the expected output fit into the business models ...
  - Customer Segments? If not, is it intended to target new customers? Is it a good or bad idea to expand the customer segments?
  - Customer Relations? Does the customers require the same services as the business's existing customers, or should new ones be introduced? If yes, is that desirable or not for the business?

  - Channels? Can the customers be reached using the same channels the business's existing customers can, or are new channels required to reach the customers of the new product? If yes, is that desirable or not for the business?
  - Key Activities? Does it require the same key activities from the business as the existing portfolio products or should new ones be introduced? If yes, is that desirable or not for the business?
  - Key Resources? Does it require the same key resources from the business as the existing portfolio of products or should new ones be introduced? If yes, is that desirable or not for the business?
  - Key Partners? Can the existing partnerships that the existing products use be used to support the new product or should new partnerships be established? If the latter, is that desirable or not for the business?

- Does the expected output fit into the business patterns?
  - Does the expected output fit into any of the business patterns that are currently attached to the business?
  - Should new business patterns be attached to the business, as a consequence of new features or possibilities brought to the business by this product? If yes, is it desirable to add those features/possibilities to the business?
  - Should any of the existing business patterns, that are currently attached to the business, be removed as a consequence of introducing the expected output of the project into the business, because it drags the business into a new direction? If yes, is that change desirable or not?

The purpose of asking those questions, is to clarify the impact that the expected output of this project can have on the business as a whole.

It should be noted that this is not relevant in all cases, though. If you are a part of a company that creates software for its customers, your product is not the outcome of the project you are working on. Your product is the development process itself, as that is what the customer is purchasing. In that case it may not be relevant to discuss many of these topics with the Customer or Product Owner, as they merely want a piece of software delivered at the end of the project that lives up to their requirements and expectations. If you are a part of a development team that is creating software for the business you work in, the steps mentioned in this section are relevant to include into the process.

While the product has previously in this report been discussed as "the product", it has up until now been called the "expected output of the project" in this section. The reason for that is, that we do not know what the product is yet. We are still discussing the expected output of this project within the given scope. Not until the initial product vision has been created, does it make any sense to talk about a product.

7.1.1 *Creating the initial vision*

When the aforementioned questions have all been answered, the Product Owner and SCRUM Master are able to create the initial product vision. The product vision was introduced in Chapter 3. This chapter introduced two different methods to create a vision: The Vision Statement 3.1 and the Vision Board 3.2. The two methods are two sides of the same coin, and the result of them are the same: A product vision.

At least one of them should be used, to create the initial product vision. The initial product vision may very well be the most important one, as that is the common ground on which the immediate future work will be based. If the initial product vision is not correct, the products foundation could also be wrong, which can have an expensive impact later in the project. Therefore, a part of this phase, is to not rush through creating the initial product vision in order to get the "real work" started. Take the time required and ensure the initial product vision clearly describes what the products is, so that it can be used to steer after in the following phases.

Once the initial product vision has been created, the part of the Project Initiation concerning Vision Development is completed and work can continue into the next phase.

The project initiation phase works with the Project view (see Section 2.3.1) from Essence [2]. The Project view works with planning, status and prioritization and represents the vision. When working in this phase, focus should therefore be on *Vision over Assignments* and creating the shared vision rather than assigning specific tasks to the project.

7.2 SPRINT PLANNING

This section will describe the second phase of the Vision Development. This phase extends the SCRUM's "Sprint Planning" ceremony and adds the required steps to include vision development. The phase is shown in Figure 7, which is a part of the Vision Development Model, shown in Figure 5.

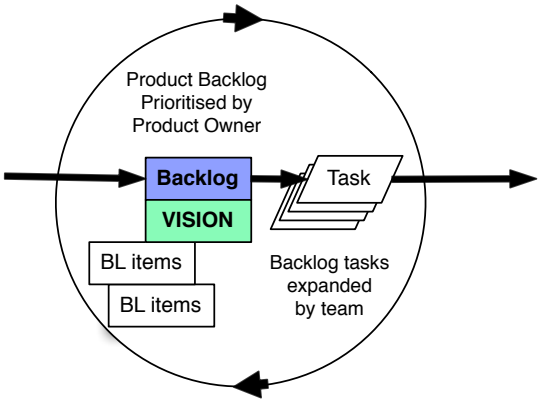


Figure 7.: Vision Development, Phase 2: Sprint Planning

From SCRUM we know that the Project Initiation provided the prioritized product backlog, containing all the items that needs to be completed in this project. SCRUM's goal of the Sprint Planning is to have two artifacts: A *sprint goal* and a *sprint backlog*. The sprint goal is a description of what the goal of this current sprint is. It can be anything from implementing a new feature to delivering a prototype of the product. It is a "mini-vision" for the current sprint and thus limited to its scope – a high level summary of the business- and technical goals of the current sprint, that the Product Owner agrees to accomplish. When creating the sprint goal, the Team and the SCRUM Master should ensure that it is in accordance with the product vision. Unlike the product vision, the sprint goal cannot be modified during the sprint, but it should be used to remember what is important this sprint. The sprint backlog is the prioritized set of items from the product backlog that is intended to be completed in this sprint. I will not discuss how SCRUM works in greater detail now, but instead use its artifacts in vision development.

7.2.1 *The Sprint Goals role in Vision Development*

The sprint goal is a description of what we plan to achieve in a sprint that must be in accordance with the product vision. If the sprint goal and the product vision does not align, one of them must be adapted to fit the other. It is often the sprint goal that will be changed to fit into the product vision since the vision is the long term goal and the sprint goal only describes what we aim to achieve in a sprint.

A number of things can happen that can cause the sprint goal and product vision to be misaligned:

- The Product Owner can change the priority of the items in the product backlog.

- The Product Owner can add new requirements to the product or change the scope of the project.
- The Team can bring new ideas forward that modifies the behavior of the product.

The Product Owner and SCRUM Master will use the Business Model Canvas to discuss whether the sprint goal is aligned with the product vision. If it is found that the product vision is not aligned with the business model, then one of them has to be updated. A number of questions should be asked to determine whether the change in the vision is desirable or not. Those questions are:

- What effect will the vision change have on the product?
- What effect will the vision change have on the business model for the product and business?
- What effect will the vision change have on the business patterns attached to the business?

7.2.2 *Sprint Planning output*

The output of the sprint planning phase is three things. First the sprint backlog, containing all the items from the product backlog that we believe can be completed in this sprint. Second the sprint goal, describing what we aim to achieve in this sprint. And finally a product vision to steer the project. The product vision may have been modified during this step if it was found necessary to move the product in a new direction, however, vision changes often happens during the Sprint Review (see Section 7.4). The question is always: *Is this modification desirable?*

The Sprint Planning phase works with the Project view and Process view from Essence (see Section 2.3.1). While we are still working with some general aspects of the project, this phase also starts to discuss the concrete product by handling the backlog. What is put into the sprint backlog should be in accordance with the vision, as it can be seen in Figure 7. If the vision has changed over time, the items in the product backlog may no longer be in accordance with the vision, thus the SCRUM Master must pay attention to this. The items added to the sprint backlog are turned into the tasks that the Team will be working with during sprint execution.

Both the Product Owner, SCRUM Master and the Team participates in the Spring Planning, taking team capacity, product backlog, business conditions, current product and available technology into consideration when creating the sprint backlog, sprint goal and product vision. Usually the product backlog is fixed at this point, but because

we are including Essence, we are trying to facilitate software innovation and thus new ideas. New ideas does not have to be made up at this point, but they should, if possible, be presented during the sprint planning so that they can be added to the backlog and included into the project. Chapter 5 presents different tools and checklists that can be used to evaluate if an idea is good or not. If a Team member presents a new idea, e.g. due to new technologies being available, it can modify the output of the sprint planning (the sprint backlog, the sprint goal and the product vision). If the product vision is modified, one needs to take the same steps as during the Project Initiation to evaluate whether the change is desirable or not.

The Sprint Planning phase works with the Paradigm and Project view from Essence (see Section 2.3.1). It focuses on how the product can be implemented (Paradigm) but also planning the forthcoming sprint (Project). Once the output of this phase is in place, sprint execution can start. As shown in Figure 7, this phase is a loop that repeats for every sprint. First comes the sprint planning, then sprint execution and finally sprint review. After the sprint review, a new sprint planning starts. Phase 2 and 3 are looped over until the end of the project. This loop continues until the end of the project when the product is ready to be delivered.

7.3 SPRINT EXECUTION

This section will describe the third phase of the Vision Development. It is built on SCRUM's "Sprint Execution" and adds the required steps to include vision development into the everyday work done during the sprint executions. The phase is shown in Figure 8, which is a part of the Vision Development Model, shown in Figure 5.

While the Vision Development Model in Figure 5 has the Sprint Execution and Sprint Review in the same phase, they will be described in separate sections.

Sprint Execution starts immediately after the Sprint Planning is done. The Team now has a prioritized Sprint Backlog of items that should be implemented, a Sprint Goal describing the goal of the current sprint and the product vision.

Sprint Execution largely follows SCRUM's description of how to execute a sprint. Every day the Team, SCRUM Master and Product Owner holds the "daily SCRUM meeting". The purpose of this meeting is to synchronize what everybody is doing, and ensure that everybody is on the right track. Usually every member of the Team answers three questions:

- What did I do yesterday?

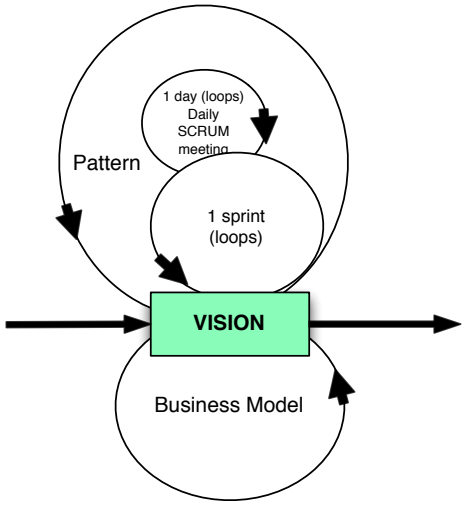


Figure 8.: Vision Development, Phase 3: Sprint Execution

- What will I do today?
- Is there anything in my way?

The purpose is not to explain the status to the SCRUM Master, but to commit to the other Team members what each individual's status is. The daily SCRUM meeting should only last for around 15 minutes. It is not intended for problem solving, but to help avoid other unnecessary meetings during the day.

During the execution of a sprint, the Team should work with two different aspects as needed in relation to vision development: Idea Evaluation and Software Innovation.

7.3.1 *Idea Evaluation & Software Innovation*

Generating new ideas is not necessarily a result of using Software Innovation, but it is a part of the reason for using it. New ideas can arise at any time, be it during development, at meetings with the customers, or even over a cup of coffee with a co-worker. When they do, it is important to be able to evaluate the ideas and decide whether to continue working with the idea. Chapter 5 presented two different ways to evaluate an idea.

The first was the Checklist for Idea Evaluation (see Section 5.1). By asking a number of questions, it can be determined whether it has been thought through and whether it will fit into the product. The second method was the SWOT analysis (see Section 5.2). A SWOT analysis is used to analyze an objects Strengths, Weaknesses, Opportunities and Threats. That should be done in relation to the product. Chapter 5 also presents a version of the SWOT analysis that includes

scoring the strengths, weaknesses, opportunities and threats. This can be used if multiple ideas should be compared and the results quantified for easier selection.

If it is found that an idea is good and that it should be implemented into the product, it should be presented during the next Sprint Planning meeting. The reason to not present and implement it immediately, is that the Sprint Backlog is already set for the current sprint, at which point it should not be modified according to SCRUM [32]. By postponing the idea presentation and evaluation, the Team can stay focused on the items in the current sprint backlog. New ideas will not necessarily change the vision. New ideas can be features that fits into the existing vision and business model. In that case it is a question of whether the resources required to implement the new features are available.

When the idea is presented, we propose that the following things can happen:

- If it is found that the idea will improve the product in some way (e.g. performance, stability, usability, efficiency etc.), then...
  - If it does not change the scope nor the vision of the project, it should be accepted. An improvement that does not require more resources and does not modify the vision should be implemented, since it will result in a better product. Otherwise...
  - If the idea changes the scope but not the vision, then it has to be accepted by the Customer and the Product Owner. Changing the scope could result in the project being late or over budget and therefore the Customer or Product Owner should accept it. Otherwise...
  - If the idea changes the vision but not the scope, then it has to be accepted by the Customer, Product Owner & SCRUM Master. Unlike a change to the scope, a change to the vision also has to be discussed with the SCRUM Master so it is incorporated into the vision and communicated to the Team. Otherwise...
  - If the idea changes both the scope and the vision of the project, then it has to be accepted by the Customer, Product Owner & SCRUM Master. Changes to both the vision and scope should be discussed by all parties to ensure that the impact on the project is understood before the change is implemented.
- Otherwise, if it is found that the idea will both add strengths but also introduce new weaknesses in the product (e.g. makes the product more stable but less intuitive to use), then...

  - If the introduced weaknesses outweigh the strengths, the idea should be discarded or redone in a way that does not introduce the weakness. It does not make sense to introduce ideas that will weaken the product, as it is en everybody's interest to deliver the best possible. Otherwise...
  - If the introduced strengths outweigh the weaknesses, the idea should be implemented if the introduced weaknesses can be mitigated in another way. Strengths should be implemented, but if they also introduce possible weaknesses, then the Team and Product Owner should seek to minimize them, if they decided to implement the idea. However, in this case, the following clause also applies:
    * If the idea changes the scope but not the vision, then it has to be accepted by the Customer and Product Owner. Otherwise...
    * If the idea changes the vision but not the scope, then it has to be accepted by the Customer, Product Owner & SCRUM Master. Otherwise...
    * If the idea changes both the scope and the vision of the project, then it has to be accepted by the Customer, Product Owner & SCRUM Master.

- Else, if it is found that the idea will mainly weaken product then the idea should be discarded if it cannot be redone to introduce more strengths than weaknesses.

The list above does not to give a complete list of questions to be asked when presenting a new idea. Instead it should put the Product Owner, SRCUM Master and the Team into the right frame of mind when working with new ideas with respect to the product vision.

The Sprint Execution works with the Product view from Essence (see Section 2.3.1). The focus is on how the response to the Challenge (the sprint goal) is implemented. Figure 8 shows that each individual sprint loops on the box representing the product vision, and how the business model and business patterns are also looping on that same vision. The loop represents the necessity of considering how a new idea can impact the business model or business patterns if implemented. The methods and questions proposed in this section helps determine whether the impact of changes to the vision is a desirable or not.

7.4 SPRINT REVIEW

This section will describe the fourth phase of the Vision Development: The Sprint Review. This phase will use the Sprint Review ceremony from SCRUM and add vision development to it. Like the Sprint Execution phase in Section 7.3, the Sprint Review uses Figure 8 to describe it.

Sprints in SCRUM are limited by time, not scope. When the Team reaches the end of a Sprint, the Sprint Review ceremony is held. Both the Scrum Master, Product Owner, the Team and other with interest, such as the customer or similar, should participate in the Sprint Review meetings. The Team will present what has been accomplished during the sprint, typically by showing some sort of demo of new features or underlaying architecture. The purpose of the meeting is not to give excuses about what has not been done or explain why something is not working as intended. The purpose is to show off what has been done. All the items in the Sprint Backlog that were not accomplished will simply be pushed to later sprints. The Sprint Review should focus on two things related to vision development: Sprint Goal and Vision Development.

7.4.1 *Sprint Goal*

The Sprint Goal was defined during the Sprint Planning, described in Section 7.2. It is the temporary goal that we are steering after during the sprint that was just finished. The Sprint Goal originates from the Vision and therefore is closely tied to it. During the Sprint Review the Team should ask themselves the following questions and evaluate the answers to the following questions:

- Did we achieve the Sprint Goal?
  - If yes, did we have too much time in this sprint to achieve the goal? - meaning that we can strive to achieve more in future sprints.
    - Did we deliver as expected?
  - If no, why did we not achieve the Sprint Goal?
    - Not enough time/resources.
    - Unforeseen technical issues.
    - Did the sprint go in another direction than the Sprint Goal and/or Vision intended?

Especially the last question is key in identifying whether we are moving in an unanticipated direction. The purpose of the questions

is to facilitate discussions that makes the team capable of making better decisions about what should happen next.

7.4.2 *Vision Development*

Figure 8 shows two loops that wraps around the vision: The Pattern and the Business Model. During the Sprint Review the participants should discuss what was found in the current sprint. By asking three questions they will be able to determine in what direction the project is going:

- Does the work that we have done in this sprint deviate from the vision?
- Does the work that we have done in this sprint deviate from the Business Model?
- Does the work that we have done in this sprint deviate from the Business Patterns?

It is important to underline that there are no definitive right or wrong answers to the three questions. It should be based on the sound judgment of the Customer and Product Owner to base the decision on reasoned and considered judgment.The purpose of the questions above, is to facilitate new ways of thinking for the parties involved in the project, so that they are aware of what they are doing and in what direction the project is heading. Only by knowing that, is it possible to identify whether the project is heading down a wrong path, before it is too late.

The first questions is: *Does the work that we have done in this sprint deviate from the vision?* Perhaps a new feature was found that could change the focus in the product, potential customers or access to the market. Perhaps a minor design change means that the product now targets either a wider or more narrow group of potential customers than earlier. Examples are given in the Case Study in Chapter 8. When it is known whether the work in this sprint is aligned with the vision, it is possible to decide whether the results are desirable or not. However, it is not possible to answer whether potential changes or deviations from the vision should be accepted before the other two questions have been answered.

The second question is: *Does the work that we have done in this sprint deviate from the Business Model?* This question represents the "Business Model" loop in Figure 8. If there is a business model for both the product and the business as a whole, they should both be taken into consideration. The business model should be compared to the vision and product at the sprint review to determine if anything has happened during the sprint, that would affect either of them.

The third question is: *Does the work that we have done in this sprint deviate from the Business Patterns?* This question represents the "Business Pattern" loop in Figure 8. The patterns are another way of describing how the business works. When new products are introduced, it could change this radically. For example when Netflix went from video delivery by mail-order to their current online streaming service, it changed both their business model and the patterns describing them. Therefore the achievements in this sprint could be compared to the patterns too, to determine if the work that has been done will affect them in any way.

When the three questions have been discussed, the Team, Product Owner, the customer, the SCRUM Master and other parties with interest in the product will have the required knowledge to determine where to go from here. The Team and Product Owner can use the Vision Pattern, presented in Section 5.3, to understand the Vision, the problems it solves and its grounds.

If it is found that nothing has been done that in any way changes or deviates from the vision, business model or business pattern, the process can continue. That is under the assumption that the original vision and business model of the product is still describing what the customer is expecting from the project.

If it is found that the work in this sprint in one or more ways changes or deviates from the vision, business model or business patterns, then it should be found out whether this change is desirable or not and ask what consequences the change will have on the:

- Vision.
- Business Model.
- Business Patterns.
- Product.

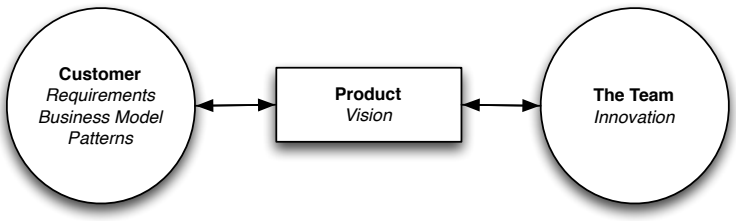


Figure 9.: Illustration of how the Customer and the Team affects the Product from two different sides

Figure 9 shows how both the Customer and the Team can affect the product and its vision. Ultimately it is the Customer and Product

Owner that decides what should be implemented, but their decisions should to some extend be based on the feedback from the Team. If the Team is able to provide valid reason for implementing something, that was not originally a part of the plans, it should be considered. That is why all parties are participating in the Sprint Review ceremony.

Along with the Sprint Planning ceremony, the Sprint Review ceremony is the time and place to discuss vision development. Input from either the Customer or the Team can affect the product, thus the Vision, Business Model and Business Patterns. As previously stated, there are no right or wrong answers to whether a specific change is desirable or not. That decision should be based on the actual situation and an evaluation of the impact it could have. If it is decided to accept a change that affects the product and the vision, it should be discussed on the next Sprint Planning how to include the new requirements into the product.

The Sprint Review phase is associated with the Paradigm view, the Project view and the Process view in Essence (see Section 2.3.1). This phase looks at what has been done and what answers have been given to the challenges (Paradigm), it discusses details close to the product and its implementation (Product) and it works with idea development and vision development, both in retrospective but also in the future (Process). When the Sprint Review ceremony is done, a new loop with Sprint Planning, Sprint Execution and Sprint Review starts. This continues until the end of the project, where all involved parties will do the Product Delivery.

7.5 PRODUCT DELIVERY

This section will describe the final phase of the Vision Development; The Product Delivery. It discusses what has been achieved compared to what was originally intended and how that affects the business model and business patterns. The phase is shown in Figure 10, which is a part of the Vision Development Model, shown in Figure 5.

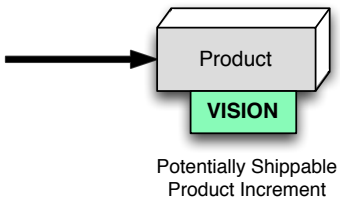


Figure 10.: Vision Development, Phase 5: Product Delivery

The product delivery is the last phase in the project. At this point, no more code will be written and no more modifications to the product or vision can happen in this project. The participants during the product delivery should use the product vision and all its reviews (changes that has been implemented during the Sprint Review ceremony during the project), the business model and the business patterns to understand and evaluate what the original intentions were, how they developed and what has been delivered. Does the product that has been delivered live up to the customers expectations? Does the product deliver what the customer needs?

If the customer is happy with the product that has been delivered, the project can be closed. Hopefully the iterative vision development process has ensured that the product was always on track with the customers wishes and requirements but also embraces software innovation (see Figure 9).

If the customer is unhappy with the product that has been delivered, a new project should be planned to implement the required changes. That project should start its Project Initiation with the final vision of this project, and add into it the feedback and requirements from the customer.

This chapter has described the theory of the Vision Development methodology and presented the Vision Development Model in Figure 5. It builds on the SCRUM development methodology and software innovation from Essence. On top of that this chapter presents a model that helps the team to ask the right question at the right time, to be sure that they are always working in the right direction. The next chapter will present a case study, where this methodology was applied during the development of a prototype of the Cloud Apps system.

8

CASE STUDY

This chapter will show how the Vision Development methodology described in Chapter 7 was used on the Cloud Apps project, that was described in the first [11] and second [13] report on this project. The concrete vision development was not described in the first two reports, and a lot of the theory described in Chapter 7 was discovered during those two reports. This chapter is therefore both a case study, but also a retrospective on the development phase of the prototype with the correct terminology in regard to vision development.

This chapter starts by explaining the development method used. Then it will describe how the project was initiated and what the foundation was, followed by a number of the Sprint Planning – Sprint Execution – Sprint Review loops, before finally looking at the result of the process and discussing how the vision has developed during the project and what effect that has had on the product.

The development of the Cloud Apps prototype was based on SCRUM and Essence. It consisted of four iterations with the purpose of delivering a working prototype of the Cloud Apps concept. That is disclosed in the second report [13]. The development process was unusual, since I was the only member of the Team and I was also the SCRUM Master.My partner in Meebox, Patrick, was Product Owner. Because I had multiple roles during the development, it was easy for me to see the potential lack of communication between the involved parties in the project. I was able to see the potential lack of communication because I could see what could easily have been taken for granted from each role, and I could see how that could lead to the project heading in a wrong direction. The online SCRUM management tool, JIRA[1] was used to keep track of the product backlog, sprint backlog and progress in each sprint.

While I was doing the work on this project, I did not realize that it was in fact vision development. It is only now that the concept has been formalized, and I want to reflect that in the case study. Therefore it will be described as if the theory were already available at the time, even though it was more of a coincidence than planned actions.

1 https://www.atlassian.com/software/jira

8.1 PROJECT INITIATION

The original idea that Meebox presented, was to create a system that could automatically create virtual servers and install & configure a web application on that server – the same functionality that give the project its name: Cloud Apps. End Customers should be able to choose an application from a list of available options, and when chosen, the system should deliver a ready-to-use application to the End Customer (notice how the users of the system are called "End Customers" – that is because Meebox is the Customer in this project, as it is them who the product will be delivered to).

The business models in the first report [11] shows the final business models that includes Hosting Providers and Application Developers. However, that was not how the first business model looked. In that model, only Meebox' public cloud system was intended for hosting, and the available applications was open-source applications such as Wordpress, Drupal, Magento etc. That made the first business model somewhat more simple than the final one (see Appendix A), as the only Customers were the End Customers, the same Channels that Meebox uses today could be used to get in contact with the End Customers and most resources were already available within Meebox.

The vision of this project that has been discussed in Section 5.4 is the final one that includes all aspects of the product. It is based on the Vision Statement (discussed in Section 3.1) that is almost identical to the Elevator Pitch [5]. If the product had had a vision during the project initiation, it would have been:

For Private clients, small and medium businesses who needs a web application to support their business, Cloud Apps is an Application-as-a-Service that allows them to choose a web-application from a catalog of available applications and have it automatically installed and configured. Unlike the competitors, Cloud Apps instances are installed on isolated server instances to secure performance, stability and data integrity. Our product differentiates by also including automatic security updates and maintenances to each Cloud Apps instance. This supports Meebox' strategy to automate as much as possible to optimize costs and earnings.

Although this vision is long, it is precise and it includes the eight elements from the Vision Statement, described in Section 3.1. The vision based on the Vision Statement as an Elevator Pitch [5]. An Elevator Pitch is a market positioning, often used as a part of marketing or presenting the idea to outsiders. It only talks about outer values of the product. Compared to a Vision Pattern Model (presented in Section 5.3), the Vision Statement does not have the same strict structure as the Vision Statement thus making it possible to look at the

internal processes of the product. The discussions about the products internal processes that the Vision Statement can provide can be used to complement the external description that the vision provides.

8.1.1 *Cloud Apps Marketplace*

One of the first things that was discussed with the Product Owner was how to manage and distribute various applications that could be installed as Cloud Apps. Other large IT companies have had success with a marketplace-inspired approach to distribute applications to users. Apple uses the Apple App Store on their mobile devices and computer operating systems. Google uses the Google Marketplace for their Android-based mobile devices. Microsoft has the Windows Market for the Windows Phone based mobile devices. That inspired us to take a similar approach to distributing applications to the End Customers. By creating a marketplace we would be able to create a simple way to show the End Customers the available applications and also add and promote new ones over time. The strengths and weaknesses of adding a marketplace to Cloud Apps is discussed using a SWOT-model shown in Table 5.

Question	Answer
Strength	Easy distribution of existing applications and promotion of new applications to the End Customers.
Weaknesses	It will add more work to implement the marketplace in the first place and more work to maintain it.
Opportunities	More sales due to easier distribution of applications.
Threats	Potential Intellectual Property issues due to the naming "Cloud Apps Marketplace".

Table 5.: SWOT analysis for adding Cloud Apps Marketplace to the product

There are more strengths than weaknesses by implementing a marketplace into Cloud Apps, according to Table 5. Therefore it was decided to implement the marketplace at this point already.

Adding a marketplace also adds new channels in the Business Model for the product, which was considered a positive consequence by the Product Owner and is therefore added to the feature requests that the product backlog was based on. At this point the vision has already been modified to include a marketplace. The required features were added to the product backlog and the Team was ready to start the development.

8.2 REPETITIVE VISION DEVELOPMENT

The development process of the prototype consisted of four sprints: Framework, back-end, front-end and testing.

The **Framework** sprint focused on implementing the most basic features of the system so that the front-end and back-end could be built on top of it. It consisted mainly of getting the foundation in place and making sure that it was ready to new features being built upon.

The **Back-end** was the actual integration with back-end systems. The first prototype only integrated with Meebox Public Cloud cloud servers for provisioning new Cloud App instances. Therefore this sprint contained functionality to integrate with Meebox' public cloud environment, and ensure that it was possible to sign into the system, create new virtual servers in Meebox' Public Cloud environment, update and delete them, etc.

The **Front-end** focused on creating the GUI *[Graphical User Interface]* that the users will be interacting with. It makes the back-end functionalities available to the End Users, and makes it possible to sign in and create new Cloud App instances using the front-end.

The **Testing** sprint was an unusual sprint in that testing is usually done at every sprint, but in this project it was chosen to do the testing after the framework, back-end features and front-end functionalities for the prototype was implemented. When the testing was done, the feedback would be turned into product backlog items and fixed during this sprint.

The result of the four sprints was the prototype that was presented in the second report [13]. The following sections will focus on the four areas that was discussed and evaluated during the development. Three of them were implemented while one was discarded. They will be described below as individual points. Compared to a regular SCRUM process, each section below would represent a sprint. The idea of implementing the features would have come up during the sprint, either by the Team due to e.g. software innovation, or it could be the Customer finding a new requirement, e.g. due to market changes. In either case, new ideas would be discussed during the sprint review. If the idea was accepted, it would be included into the product backlog during the next sprint planning. If not, it would be discarded during the sprint review. The sections below presents the discussions that would have taken place during a sprint review and shows the thoughts and considerations that were made when looking at these ideas.

8.2.1 *Hosting Providers*

One of the first requirements to create a new Cloud Apps instance, is to have a hosting environment in which the virtual server hosting the Cloud App can be provisioned. The original idea was to only use Meebox' public cloud environment and provision all new instances there. Then the idea emerged from the Customer: What if the system could support any public cloud environment? The vision would have to be modified to include:

> *(...) Unlike the competitors, Cloud Apps instances are installed on isolated server instances to secure performance, stability and data integrity.* ***The instances can be provisioned in any cloud environment that is a member of the Cloud Apps Hosting Provider Network.*** *(...)*

If other hosting providers makes their cloud environment available for new Cloud Apps instances to be provisioned on, they would in fact be customers with Meebox. That would change the business model so that, instead of the End Customers being the only customers, the Hosting Providers would also be customers. That creates a multi-sided business model where the End Customers and the Hosting Providers depend on each other. This can be seen in Appendix A where the Hosting Providers' role is shown in Blue. Table 6 shows the SWOT analysis of adding multiple Hosting Providers to the system.

Based on the SWOT analysis in Table 6 and the impact, that the change will have on the Business Model, it was decided that Cloud Apps should support multiple Hosting Providers.The hosting providers should pay a setup-fee to be a part of the platform, and then End Customers will be able to choose their cloud hosting environment when provisioning new Cloud App instances. Meebox may make less money when an End Customer chooses another Hosting Provider than Meebox, but the ability to choose a provider in another country may be the final argument that makes the End Customer choose a Cloud App rather than an alternative, so it is considered a feature that strengthens the product.

8.2.2 *Application Developers*

The vision states that various open-source web-applications should be offered as Cloud Apps through the Cloud Apps marketplace. During the second sprint, when the Team was implementing the Cloud Apps marketplace, it was discussed what kind of applications should be offered on the marketplace. At that point the Team suggested: What if developers were able to create applications specific to the Cloud Apps marketplace and the Cloud Apps platform, in the same way

Question	Answer
Strength	Better value proposition because we "automatically" helps the Hosting Providers acquiring new clients. The system is more accessible due to multiple cloud environments being available, and finally there is a positive branding effect that the system supports multiple cloud environments, e.g. some of the international leaders like Amazon, Google or Rackspace.
Weaknesses	It will require more work to be able to implement support for multiple Hosting Providers that may be using different back-end systems and therefore require individual implementations.
Opportunities	More sales because Cloud Apps instances can be installed on multiple cloud environments around the world with different high-profile companies. Also an opportunity to give the product positive branding when using acknowledged Hosting Providers.
Threats	Meebox may make less money on hosting because the Hosting Providers will need a share of the income from the End Customers.

Table 6.: SWOT analysis for adding multiple Hosting Providers to the Cloud Apps product

they can do it for Apple's App Store and Google Play (the Android marketplace)?

Allowing Application Developers to create custom items for the marketplace is another type of modification than adding the Hosting Providers. Therefore it was evaluated using the Checklist for Idea Evaluation (see Section 5.1). The result of this can be seen in Appendix C. The addition of Application Developers is shown in the Business Model in Appendix A, displayed in Green.

It is believed that having third party Application Developers on the platform will give a new dimension to the platform, because it will add applications to the marketplace that may not otherwise have been available. That can give a competitive edge towards the End Customers that will make them choose our system.

During the following Sprint Review, the vision had to be modified to include the new Application Developers aspect of the system. The vision is modified in the following way:

(...) Cloud Apps is an Application-as-a-Service that allows them to choose a web-application from a catalog of available applications and have it automatically installed and configured. ***The catalog includes custom designed applications, made by 3rd party Application Developers, specifically for the Cloud Apps platform.*** *(...)*

The addition of Application Developers modifies both the Vision, the Business Model (see Appendix C – green text) and the Business Patterns. Instead of only providing applications to End Customers via the marketplace and being a Hosting Provider, Meebox now also has to facilitate development environment for the Application Developers and handle additional license fees that the Application Developers may charge. Therefore it also modifies the patterns that applies to the business because it has to include these new aspects.

8.2.3 *Customization Services*

After the Hosting Providers and Application Developers were added into the vision and product backlog, it was suggested by the Customer to add customization services to the product. Meebox should offer various customizations to the End Customer, including setup, custom development, custom design and other tasks that the End Customer wished to customize the Cloud App instance to their specific needs. The idea is analyzed with a SWOT-analysis, shown in Table 7.

It was decided that adding customization services to the product would deviate too much from the original vision and drag the product in an undesired direction. It would add a lot of manual labor to the team, that would have to assist End Customers with setting up, configuring and designing their Cloud Apps instances to their needs. The original vision, the product business model and Meebox' business patterns focuses on automation. Therefore "Customization Services" was discarded as it goes against these values, and because it is believed that it will have a stronger negative than positive impact.

8.2.4 *Automation*

Although Meebox does not have an interest in involving themselves too much in the end customers Cloud Apps instances (e.g. by turning down the suggestion about Customization Services in Section 8.2.3), Meebox has a natural interest in ensuring that the Cloud Apps instances are always online and available. If they are not online or not performing as expected, the End Customers cannot use them which can result in unhappy customers or loss of customers. There can be many reasons why an application is not available, anything from network outage or capacity overload to software errors. Two of the pos-

Question	Answer
Strength	By offering customization services to the End Customer we may attract customers with little or no technical experience because we can assist them in setting up and configuring their Cloud App instance. It may also have a positive effect on the brand because it will signal a high level of customer service.
Weaknesses	It will require a lot of manual labor to offer customization services to the End Customers and it will remove focus from automation.
Opportunities	By offering customization services it is likely that Meebox' team will get a deeper insight into the available applications and make them better prepared to help and support the End Customers.
Threats	There is a risk that it may remove so much focus from automation and require too many resources, so that it will keep the business back in other ways.

Table 7.: SWOT analysis for adding customized services to the Cloud Apps product

sible reasons for unavailability are capacity overload and out-of-date software. They were suggested by the Product Owner during the development, and the discussion that happened during the Sprint Review are in the following two sections.

8.2.4.1 *Automatic application updates*

The idea of automatically being able to update the web application that is being installed in a Cloud App instance, e.g. a Wordpress or similar, is in itself a good idea. It ensures that the End Customer always has the newest version of the software, including all performance-, function- and security updates. However, there are also some cons that should be considered:

- **Ethics in modifying customers data.** It may not be all customers who are happy with Meebox updating their applications automatically for principal reasons. Some customers feel strong about their data, so if the idea is implemented, perhaps it should include an "opt-out" option, so that the customer can choose that their applications should not be updated.
- **Customizations may break with automatic updates.** If the customer has installed custom plugins or made other customiza-

tions to the web application, they may break as a consequence of automatic updates, since they can introduce new, modify existing or remove old functions. A solution could be to offer the customers an upgrade and let the customer push an "upgrade"-button. In that way, they also accept that the update will be installed.

On top of that is the technical implementation of automatic updates. The system must support update procedures that may be unique to each application in the Marketplace, and the resources required to implement and maintain this should also be taken into consideration.

In some cases, automatic application updates will be a good idea, since it will help the customer to ensure that their Cloud App instance is always up-to-date. In other cases it will not be a good idea, since it may break the customers application if he has installed customizations. The solution could be a default automatic update enabled, but give the customers an "opt-out" that will allow them to choose not to have their applications automatically updated.

8.2.4.2 *Automatic resource scaling*

If a web site gets too many visitors at once, it is not uncommon that it becomes unavailable because the servers are not able to handle the amount of visitors. A solution to this is to buy a server that has enough resources to handle peak time load. However, this is also an expensive solution, since in most cases, the large amount of available resources are not required most of the time, but the customer must at all times pay for the resources that are allocated. To solve this, the Product Owner suggested automatic resource scaling according to real time requirements. That would give two things:

- **Only allocate the resource required**. Automatic resource scaling is both up and down. When more resources are required, they will be allocated, and whey they are not needed any more, they will be removed. This means that any Cloud App instance will have the resources required to do its job at any given time.
- **Flexible billing**. The End Customer pays for the resources that are allocated to a Cloud App instance. By implementing dynamic resource allocation, it ensures that the application is always available and that the End Customer only pays for the resources that are actually required to keep it running.

It was made clear by the Team that automatic resource scaling is a large task that cannot be included into the backlog of the prototype. However, it was agreed that it should be implemented into the production version of the Cloud Apps system.

Automating as much as possible fits into Meebox' existing business model, and their internal slogan is *"Handle by exception, and exception only!"*. The downsides of implementing this kind of automation were difficult to find. In the case where the automatic application updates were not implemented correctly, it could potentially destroy many End Customers Cloud App instances, so testing and verification is especially important when it comes to automation, because it modifies production environments.

It was decided to include automation into the Product Vision which modifies it in the following way:

(...) Unlike the competitors, Cloud Apps instances are installed on isolated server instances to secure performance, stability and data integrity. The instances can be provisioned in any cloud environment that is a member of the Cloud Apps Hosting Provider Network. ***Each Cloud App instances can automatically update the installed application and allocate or remove resources on demand.*** *(...)*

8.3 PRODUCT DELIVERY

The Product Delivery is the final ceremony in the project. It follows almost the same procedure as the Sprint Review, except instead of discussing the next sprint, it discusses potential future work. In the Vision Development Model, shown in Figure 5, it can be seen that the final phase (Product Review) only contains the Product and the Vision. During the development and the individual sprints, the vision was modified several times. Four suggestions where brought up, out of which three were accepted:

- Hosting Providers.
- Application Developers.
- Automation.

The original vision that was agreed upon by the Customer, Product Owner, SCRUM Master and the Team at the beginning of the project was this:

For Private clients, small and medium businesses who needs a web application to support their business, Cloud Apps is an Application-as-a-Service that allows them to choose a web-application from a catalog of available applications and have it automatically installed and configured. Unlike the competitors, Cloud Apps instances are installed on isolated server instances to secure performance, stability and data integrity. Our product differentiates by also including automatic security updates and maintenances to each

Cloud Apps instance. This supports Meebox' strategy to automate as much as possible to optimize costs and earnings.

The vision grew during the development and came to include more elements as new ideas rose and got accepted:

For Private clients, small and medium businesses who needs a web application to support their business, Cloud Apps is an Application-as-a-Service that allows them to choose a web-application from a catalog of available applications and have it automatically installed and configured. The catalog includes custom designed applications, made by 3rd party Application Developers, specifically for the Cloud Apps platform. Unlike the competitors, Cloud Apps instances are installed on isolated server instances to secure performance, stability and data integrity. The instances can be provisioned in any cloud environment that is a member of the Cloud Apps Hosting Provider Network. Each Cloud App instances can automatically update the installed application and allocate or remove resources on demand. Our product differentiates by also including automatic security updates and maintenances to each Cloud Apps instance. This supports Meebox' strategy to automate as much as possible to optimize costs and earnings.

The final vision is longer than I would normally suggest for a product vision. Usually, a vision should be kept short, unambiguous and precise and not include details concerning the implementation, to ensure that everybody working on the project can easily understand it and use it to guide their work without limiting their creativity.

The reason that the vision for this project grew so large, is because the project is rather complex and contains many different aspects that should be included into the vision to give the full picture of what the product is, what it can do and the problems it solves for the customers.

8.3.1 *Final vision compared to Meebox' business model*

Section 4.2 defined the 6 of the 55 Business Patterns that applies to Meebox' business. They were:

- Pattern 6: Cash Machine. The customer pays up front, resulting in increased liquidity.
- Pattern 20: Guaranteed Availability. The service must always be available to the End Customer.
- Pattern 34: Orchestrator. Meebox focuses on its core competences and outsources everything else.
- Pattern 35: Pay per use. The End Customer pays for their exact usage only.

- Pattern 49: Super Market. Selling a large variety of available product under one roof.
- Pattern 52: Two sided market. Having two or more customers depending on each other to make business.

This section will discuss how the final vision fits into the applied Business Patterns and if any should be added or removed. The discussions are concrete to this case study and may not apply in other situations.

Cash Machine. The End Customers are paying up front for each Cloud App instance. Their usage is estimated and future payments are based on past usage history to combine pay-per-usage and up-front payment. Hosting Providers is charged an up-front fee to join the platform. Application Developers can join free of charge but are charged a percentage fee of any licenses they may charge the End Customers.

Guaranteed Availability. Cloud Apps is essentially a hosting product like Meebox' other products, thus availability is still important. If a Cloud App instance is not available then the End Customers cannot use it; therefore guaranteed availability also fits Cloud Apps.

Orchestrator. With Cloud Apps, Meebox focuses on hosting, which is its core competences. Development of applications is out-sourced to Application Developers and parts of the core of Cloud Apps is developed by other companies and licensed by Meebox. By also having multiple Hosting Providers and the Cloud Apps Marketplace, Meebox orchestrates and facilitates the business between several different parties to fulfill a need with the End Customer.

Pay per Use. The resource usage of each Cloud App instance is measured and billed accordingly. A core principle in the system is pay per use to ensure that an End Customer only pays for his actual usage at any point in time.

Super Market. This pattern does not fit Meebox' regular business. Meebox usually sells a small variety of hosting products, but with the introduction of the Cloud Apps Marketplace, each application in the Marketplace is a product in itself. They are sold cheap to compete on the market, and they are all available at the same place. Therefore the Super Market-pattern has been added to describe Meebox' business.

Two sided market. Usually Meebox offers hosting from its own data centers and the customers provide the software that they want hosted themselves. With the introduction of Cloud Apps, three different customers are now needed to sell the product, and the customers are all depending on each other. To get a head start, Meebox will offer its own cloud environment as a hosting partner and add a number of open-source applications to the Marketplace. Because there are now multiple customers that are depending on each other, the two-sided

market-pattern has been added to describe Meebox' business.

No patterns were removed since the core product is still the same; Hosting.

8.3.2 *Impact of Vision Development*

The main question asked in Chapter 7 was: *Is this change desirable or not?* The answer is of course a subjective evaluation of the individual situation. *Do I believe that this change is desirable or not?* Chapter 7 contains examples of what to do in both situations. That leads to the question that was asked in every Sprint Review in the case study, and a question that should also be asked now during the Product Delivery: *Are the changes that the vision has undergone desirable or not?*

In this case study, the Customer and Product Owner was without doubt – they were desirable. Although the vision is broader and contains more aspects now than at the beginning, the core idea is the same. In this case, the core idea has not been modified and the project has not been pulled in a radically different direction than was originally intended. In the Customer's and Product Owner's opinion the product vision is stronger now and the result will be a better product with a better competitive edge on the market that will hopefully result in more sales.

What would have happened without Vision Development? Some of the same ideas may still have been brought up and discussed, but the lack of a defined process to handle new ideas and changes in the vision could have resulted in different scenarios:

- Ideas were discussed during a Sprint and forgotten at the Sprint Review, thus never approved by the Product Owner and added to the Product Backlog.
- Discussions during the Sprint Review result in approved ideas and change requests that are added to the Product Backlog but poorly communicated to the Team. Therefore they have deviating views on what product they are creating, resulting in waste of resources during development on meetings and synchronizing the Team.
- Ideas are approved without the Product Owner or the Team fully understanding what it actually means. What impact will this new idea or feature have on the product and/or on the Customer's business?
- Ideas are disapproved without the Product Owner or the Team fully understanding what it actually means. What impact will

the lack of this new idea or feature have on the product and/or on the Customer's business?

- New ideas introduced by the Product Owner are added to the backlog during a Sprint Review or Sprint Planning meeting, but the Team is unsure of the impact it will have on the code that has already been written, which can potentially end up in either the Team delivering something other than what the Customer wants, or that the Team needs more time to finish the product.

These scenarios were among the ones avoided and instead new ideas were discussed during the Sprint Review ceremonies as described in Section 8.2 where three ideas were approved and one was discarded based on thorough analysis of each idea, its impact on the product and the business. This was combined with using a Business Model Canvas to analyze the impact of new ideas. The impact of each of the three customer types in the multi sided business model (End Users = red, Application Developers = green and Hosting Providers = blue) is shown in products Business Model Canvas in Appendix A.

The purpose of this chapter was to show how the Vision Development methodology described in Chapter 7 was used in practice on the Cloud Apps products, and what was gained from using that compared to a regular SCRUM or Software Innovation process. New ideas from the Customer, the Product Owner (business-related ideas) and the Team (technology related ideas) were analyzed and, if approved, the vision was adapted to embrace the new features that the ideas described. This gave all parties in the project a continuous and always up-to-date vision to steer after in both the immediate and long-term work.

Part III

EVALUATION

Part III is the last in this report, used to evaluate what has been found in the previous parts and chapters. First, the discussion in Chapter 9 addresses different aspects that could have been done differently, what that would have meant for the project and what direction it could have been taken in. The conclusion in Chapter 10 expands on the validity of the hypothesis from Section 1.4.1 and discusses the work in this report. Finally, future work in Chapter 11 discusses possible future work on the Vision Development methodology.

9 DISCUSSION

The Vision Development methodology and the Vision Development Model in Figure 5 is a combination of a number of existing tools, methods and methodologies that helps bring structure to vision development during a development process. The purpose is to provide all parties involved in a project with a clear vision that they can use as a guideline in their work, but perhaps more importantly, to provide them with the necessary tools to identify when changes require the vision to be modified and how to handle such modifications. This chapter discusses the findings, what could have been done differently, and what effect that would have had. It discusses how the findings in this report can be used by the involved parties in a project, and finally how it can be used in other contexts than the one shown in the Case Study in Chapter 8.

ELEMENTS IN THE VISION DEVELOPMENT METHODOLOGY

Vision Development is built on top of SCRUM and Essence. I chose to build on an iterative development method, because I believe that a vision should mature over time, and at the same pace as the development of the product, to ensure that the vision and the progress of the project is aligned with the customers wishes and requirements. By building on top of SCRUM, an already widely adapted development method, I was able to re-use many of the ceremonies and add new items to them to handle Vision Development. Having SCRUM as a foundation will hopefully make it easier to implement into everyday routines, as it will build on top of something familiar, compared to a model that builds on either a less used- or brand new development method.

Software Innovation is, to some extent, facilitation of product innovation of software-based products.Product Innovation can often be achieved by tuning one of the four "What, Who, How and Value" parameters (see Section 4.1 for reference). Product Innovation can be by inventing something new and using new technologies (the What?), changing the billing model (the Why?), by restructure how the product works internally and increase profits (the How?) or by automating tedious tasks to optimize costs and improve customer usability (the Value?). Essence works with the same views and values as the "What, Who, How and Value" model (although they are named and structured differently in Essence) and aims to extend SCRUM with values, views and roles to facilitate software innovation during the devel-

opment process (see Section 2.3). The reason I chose to use Essence instead of e.g. the "Eight work-style heuristics for creative system developers" by Jeremy Rose [1] that also works with Software Innovation in creative development environments, is that Essence is closely linked to SCRUM and once again uses familiar tools and processes. I wanted to create something that was easy to implement in the daily routines.

Vision Development adds new points to the agenda at the SCRUM ceremonies. During these ceremonies, new ideas, suggestions or modifications are discussed and it is decided whether or not it is desirable and should be implemented. To evaluate whether an idea was good or not, SWOT analysis and Idea Evaluation methods was introduced in Chapter 5. I chose to use simple tools for quick idea evaluation to make it easy to gain an overview of whether an idea is good or not in its context. However, I believe that one must fully understand the impact of new ideas before it can be decided whether to implement them. That includes all aspects from short-term development costs to long-term impact on the customer's business. To get the overview of the impact of a decision, Vision Development uses two tools to analyze the product vision: The Business Model Canvas (see Chapter 1.2) and Business Patterns (see Chapter 4). The Business Patterns describe the patterns that the business acts by, using the "What, Who, How and Value". However, in some cases, it is not enough to be able to describe a business with those four views, so the Business Model Canvas was introduced to give a deeper and more thorough understanding of the business using a business model. By implementing new ideas and potential changes into the models, the customer and product owners are able to see the effect before using resources implementing them.

The theories described above all leads up to the Vision Development methodology, and how to handle Vision Development during a development process. Chapter 3 describes how to create the initial vision for the product that is the basis in the Project Initiation ceremony. Figure 5 shows the Vision Development Model that is described in Chapter 7. The model shows how we believe that a team can maintain a product vision during the project.

It is difficult to say what would have happened, if we had used another development method, or other tools, as foundation for the Vision Development methodology. Since it is a methodology and not a method or tool, the final result would probably be the same for the team using vision development. They would have some way of facilitating vision development to give them a shared goal to aim for during development. Although the vision may change several times

during the project, all parties know that what they are currently aiming for is what is correct right now. If the product vision changes in the future, the project is prepared to embrace that change.

PERSPECTIVES AND ABSTRACTIONS

Vision Development can be seen from three different types of abstractions: The Business, the Project and the Product (reference to Chapter 6). At the same time it can be observed from four different perspectives: The Customer, the Product Owner, the SCRUM Master and the Team (reference to Chapter 7). Vision Development needs different perspectives and abstraction levels to take full advantage of the different tools for idea evaluation, to ensure that the full impact of implementing new ideas is understood. The perspectives and abstraction levels are there so that the Team and the Customer fully understands the impact of a decision, and makes this decision with all the necessary information being available. The addition of Hosting Providers to the Cloud Apps product is a good example of this (see Section 8.2.1). From the Team's point of view, it may not be a huge technical challenge to implement support for multiple hosting providers. They would of course have to use extra resources to create API's to interface with each of the Hosting Providers, however, that is a one-time cost per Hosting Provider. While it may not be a problem from the Team's point of view, and therefore not something they will pay much attention to, adding multiple Hosting Providers to the product can prove to be a big competitive advantage on the market, and therefore it is important to the Product Owner and Customer, when looking at it from the Business-abstraction (as shown in the Business Model Canvas in Appendix A).

This report has not given much attention to the four roles in Essence, the Child, the Challenger, the Responder and the Anchor (see Section 2.3.2). They should be used by the Team during the sprint execution to discuss the challenges from different points of view, and to facilitate innovation and creation of new ideas. While I recommend that the Team uses the four roles during a sprint execution, it focuses more on facilitating idea creation than idea evaluation and therefore it has not been given much attention, although it is important since it is one of the methods for creating the ideas that vision development in a project can be based on.

CASE STUDY DISCUSSION

The Case Study presented in Chapter 8 was an obvious choice, since the Cloud Apps project is what I have been working with, and it is where the need for vision development originally emerged. However,

during my work with Vision Development and the Vision Development Model I have discovered that Cloud Apps is perhaps not the most suited project for a case study. In the same way that there are no right or wrong answers to whether a new idea is desirable or not (it depends entirely on the context in which it is suggested and the people making the decision), there is not necessarily a right or wrong way to use Vision Development.

The case study has the obvious flaw of me having too many roles in the project. I worked both as SCRUM Master, as the Team and my business partner from Meebox was Customer and Product Owner. Normally you would have different people in the different roles, sometimes more than one Product Owner and definitely a team consisting of more than one developer. Therefore I believe that the following list has to be taken into consideration when reading the Case Study:

- Pros:
  - Because I had so many roles in the project, it was easy to fully take advantage of Vision Development, since I had full understanding of how it was supposed to be used.
  - The need for vision development originally emerged with me during the Cloud Apps project, and it was here it was first tested in practice.
- Cons:
  - The creativity was limited to two people during the development (the Product Owner and me).
  - The product has not yet made it into production so there is no feedback from the End Customers, Hosting Providers and Application Developers.
  - The project only had four sprints. A longer project with more sprints would have allowed for more time in which the vision could have developed.

Me having multiple roles during the project could to some extend have been supported by the Roles in Essence (see Section 2.3.2). By putting myself in different roles in different situations, discussions would have been conducted from different points of view. If you are able to actively take on the Role, it would mitigate the downsides of not having an entire team to discuss ideas with, although it is not the same because different people provides individual feedback regardless of their role.

The case study does not test all combinations of idea evaluation and vision development described in the Vision Development methodology. It does not discuss the impact of new technologies that can be

implemented or how the outside-market can affect the project, because it is not relevant in this case. To get a better understanding of how the Vision Development works in practice and to keep developing the methodology itself, it should be tested on more projects with more people working on it. The methodology should be tested on projects with time and resources allocated, and projects that are being executed in organizations that are ready to embrace change.

VISION DEVELOPMENT IN OTHER CONTEXTS

In what contexts, other than the one described in this report, could Vision Development be used, and what changes or modifications would be required to make it usable in those contexts? Some examples and thoughts about potential modifications are listed below. They all have in common that they are software development projects.

- Large internal development project with multiple development teams. If a project is so big, that it requires multiple development teams and uses e.g. SCRUM of SCRUMS, then the Vision Development methodology would have to be modified accordingly to embrace the changes in the development method. During the SCRUM of SCRUMS meetings, additional steps should be added to handle vision development. These steps would likely be much like the existing Sprint Review ceremony. The challenge would lie in communicating ideas from the individual team member to the SCRUM of SCRUMS meetings and afterwards have them accepted and implemented.
- A project that is not an internal project, but something that is being developed on behalf of a customer, meaning that the products customer is not the same as the project customer. This would require additional communication between the developers and the product owners to ensure that the visions are aligned. In that case, the developers also has to keep their own business model and visions in mind, as their value proposition is not what the product offers, but the development of the product itself.
- Projects with very frequent change requests from the Customer or Product Owner. How would the Vision Development methodology handle a project where the Product Owner or Customer continuously added new feature requests or scope changes to the product? Would the vision be able to be up-to-date at all times, so that everyone knows what they are doing? In this case the SCRUM master would probably initiate shorter sprints so that there are more Sprint Planning and Sprint Review meetings where the Customer and Product Owner participates, to

give opportunities to align expectations to the project. Those ceremonies would also serve as the time where the vision and potential changes visions would be discussed.

There are of course many other scenarios where Vision Development should be tested. Testing the methodology in the scenarios described above will provide valuable information about how it works on different projects and help develop the methodology.

10 CONCLUSION

The purpose of this report is to show and describe the Vision Development Model and methodology that I have used during the development of the Cloud Apps project to manage the vision and ensure, that the development was heading in the right direction at all times. To do this, I asked nine research questions in Section 1.4 that should help provide the necessary information to investigate the validity of the hypothesis in Section 1.4.1. In this chapter I will first answer the research questions, then expand on the validity of the hypothesis and finally discuss the outcome of this report.

The research questions were separated into three categories: Vision Change, Change Impact and Available/relevant tools and methods. The questions are answered below.

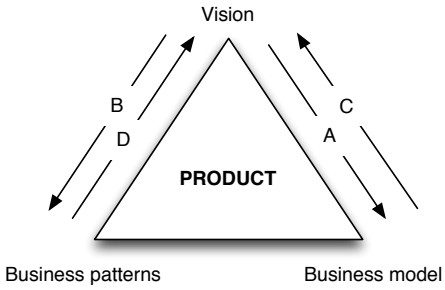


Figure 11.: How the Vision affects the business model and business patterns and vice versa.

How do we know if the vision has changed?
A change in the vision only comes unannounced if you are not aware of what is happening during the development. Vision changes can emerge both internally within the Team (e.g. through software innovation, new ideas, features or solutions) or externally from the Customer or Product Owner (e.g. through new feature requests, changes in the market, customer segments etc.). By analyzing the input and comparing it to the current vision, the business model and business patterns, one is able to see whether what we have done, or what we should do in the future, deviates from the current vision.

How do we handle a vision change?
When it has been agreed to change the vision, it needs to be communicated to everyone working on the project. The SCRUM Master

must ensure that everybody understands the new vision, and adapts their future work accordingly. If the whole team is not working with the same vision, their work can deviate and resources will be wasted.

How can a change in the vision affect the business model?
See Figure 11 arrow A. If the vision changes, it will usually also mean that the outcome of the project, the final product, will be different from what was originally planned. If the product is different from what was originally intended, it can affect the business model. How the deviation affects the business model depends on the business model and the changes in the product. It could be any parameter in the business model e.g. the customer segments, cost structure or maintenance requirements. The specific change in the business model is not important to the methodologies – it is important to identify the change and react to it.

How can a change in the vision affect the business patterns?
See Figure 11 arrow B. The business patterns are another way of describing a business. In the same way that a change in the vision can modify the business model, it can modify the business patterns.

How can a change in the Business Model affect the vision?
See Figure 11 arrow C. Vision development and changes in the vision can originate both internally in the Team during development and externally from the Customer or Product Owner. In the latter case, outside factors can affect the Customer's business model. In changing the business model it is possible that the original product no longer fits into the business. It will therefore have to be adapted to the new business model, and this also includes a vision change so that the vision reflects the modified product that the customer wants.

How can a change in the Business patterns affect the vision?
See Figure 11 arrow D. Changes in the business model often leads changes in the business patterns and vice versa. A change in the business patterns may lead to changes to the vision and the product, in the same way that a change in the business model can.

What tools and methods are required to evaluate a vision and idea?
To evaluate whether an idea is good or not I introduced a number of tools: The SWOT analysis and a checklist for idea evaluation for evaluating single ideas and the SWOT analysis with points and the NAF-analysis for composite ideas. A vision should consists of a number of elements to ensure that it captures all the required aspects, to give a full and meaningful description of the product. The presented tools enables us to find meaningful ideas and visions that are thought

through. Three tools, the Vision Statement, Vision Board and Vision Pattern, was presented to create, discuss and mature a vision during the project. By comparing the vision to the business model and business patterns, the Customer and Product Owner can determine whether the impact of the change is desirable or not to their business.

How can we facilitate a development phase that can embrace changes to the vision?

The Vision Development Model presented in Figure 5 and described in Chapter 7 captures how SCRUM and Essence can be adapted to include vision development. Vision Development describes what to do in the different phases in SCRUM: Project initiation, sprint planning, sprint execution, sprint review and product delivery and how to analyze ideas, make them into vision changes and understand the impact of such a change.

How can vision development include software innovation, and how do they affect each other?

Software innovation is a natural part of vision development because input and new ideas can originate from the Team too. Some of those ideas can originate from using the methods and tools described in Essence. On the other hand, using a vision as a common goal to aim for during the development, may lock some people in a tunnel vision, where they become so fixated on the vision, that they stop being innovative, which should of course be avoided. Using Vision Development should not prevent the Team from being innovative during the development process. It is amongst others analyses like the ones described above and shown in Figure 11 that helps to identify if the vision, business model and business patterns are not balanced.

Using the knowledge from the answers to the research questions and the knowledge gained in this report, it is possible to expand on the validity of the hypothesis. From Section 1.4.1 it stated:

Adapting a known development method to include vision development will enable the customer, developers, product owners and managers to actively see, understand and work with a product vision, and use it as a common goal to aim for during the development of the product, to ensure that it is heading in the desired direction. It will also identify both inner and outer changes to the vision, and based on that, enable the customer and product owner to make intelligent decisions on whether the change in the vision is desirable or not.

Vision Development is using SCRUM to have an iterative foundation. Therefore the Vision Development methodology can be built naturally on top of existing SCRUM ceremonies. It adds new questions

and facilitates discussion at the ceremonies, that include all parties in the project to identify if what we are currently doing is what we want to do, and to identify if what we think we want to do, is actually what we want to do. Finally the Vision Development methodology create a shared and common vision that everybody can use as a common goal to aim for during the development of the product. A vision is valid until it is replaced by a new vision. A new vision can be introduced during the Sprint Review ceremony where the parties in the project discusses the achievements of the finished sprint, what should be developed in the next sprint, how that fits into the business and if any outside factors has come into play that affects the vision and the product.

When something new has been approved and is to be a part of the product, the full impact on the product and the business should be understood before the new feature is added to the product backlog. This is achieved by investigating how the change in the vision affects the business model of both the product and the business, but also the business patterns. It is not the purpose of the methodology to tell whether a specific change in the vision is desirable or not. That decision is unique in every case, but by following the Vision Development methodology the Customers and Product Owners are able to make the decision while being aware of its impact on the product and the business.

11

FUTURE WORK

This report presents the Vision Development Model in its initial version. It has been tested on a single project, and it is argued why using the Vision Development methodology will aid all parties involved in a project in ensuring that they are on the right track.

Future work on the Vision Development methodology depends on the outcome of it being tested in different scenarios. I imagine four cases were I would see it beneficial to have it tested:

- A project with a *small* scope and a *small* team.
- A project with a *large* scope and a *small* team.
- A project with a *small* scope and a *large* team.
- A project with a *large* scope and a *large* team.

The purpose is to capture how the teams integrate vision development into their development process, and if it helps them to stay on track and align expectations and results with the customer. Based on their feedback, the methodology can be improved based on the experiences gained by using it. Changes to the methodology should of course be based on a sufficient set of data. I do not believe that one project of each type is sufficient. Instead I would recommend testing the existing methodology on multiple projects, capture the effects and evaluate the results. An interesting experiment could be to use elements from the methodology itself when developing it – to make sure, that the development of the methodology stays on track.

When developing the methodology, it could also be interesting to see if other theories can be integrated. Osterwalder and Pigneur's Business Model Generation [24] also discusses how Blue Ocean by W. Chan Kim and Renee Mauborgne can be used to take the existing business model and move it from an existing competitive red ocean into a new and undiscovered blue ocean without competition, by only modifying small elements in the business model. If incorporated into the Vision Development methodology, the Blue Ocean theory could add a two-way effect, where the methodology also helps to develop the business model, as Vision Development can help identify new blue oceans that a product can target.

Future work should be centered around putting the methodology to use, capture the experiences and evaluate whether they should be integrated to improve the methodology.

Part IV

APPENDIX

A

CLOUD APPS BUSINESS MODEL

The business model for the Cloud Apps product, introduced in the first report in this project [11].

Cloud Apps is based on a multi-sided business model with three customers:

- End customers (red): They will buy and use the Cloud Apps.
- Application developers (green): They will create new applications for the platform.
- Hosting providers (blue): They will provide a Cloud Server platform on which new Cloud App instances can be provisioned.

The value propositions for the end customers are very important, as it primarily is the end customers that needs to be impressed for this system to succeed. It is the end customers paying for the entire system, so it needs to meet - if not exceed - their expectations.

BM-Prototype 3

Entrepreneur-project ____
Intrapreneur-project _X___

The Business Model Canvas

Designed for: Total. End User, Developer, Hosting Provider

Designed by: Anders Eiler

On: *Iteration:*

Key Partners

Application Developers.

Hosting Providers.

Key Activities

Platform maintenance and development.

Marketing, selling new Cloud Apps to end users.

Marketplace updates.

Key Resources

Cloud Apps Platform and marketplace.

Value Proposition

Easy application deployment.
Data integrity. Good performance.
Automatic updates and scaling.
Customization to client needs.
Usability - simple application management. Price – cheap compared to fully managed solutions.
Newness – marketplace through which their application can easily be deployed. An easy way to sell more. Availability.
Newness – development platform which provides an online hosted environment on which the developers can develop their applications.
Newness: New clients "automatically", more business.
Accessibility: Keep the platform running, we'll provide the clients.
Brand: Hosting providers get to brand their solution in the marketplace.

Customer Relationships

Self-service (order), Community (daily) & Personal Assistance (support).

Self-service for most, community for dev. Questions.

Personal assistance.

Channels

Awareness –SEO+adwords.
Active presence on SM/Forums.
Purchase – through our website.
After sales – push current clients to the new platform.
Awareness in Development forums etc.
Awareness: SM Presence + SEO.
Personal sales: Few Hosting providers. Acc. manager

Customer Segments

Mass Market (generic applications).
Niche Market (specific applications).
Small/Medium business & Entrepreneurs.

Niche market: Application developers who want to publish their home-brewed web-applications.
Entrepreneurs/small-medium business.

Niche market: Cloud Hosting Providers who can offer a platform on which the Cloud Apps can be provisioned. This can e.g. be Amazon, Rackspace, nCloud etc.

Multisided platform (end users, application developers, hosting-providers)

Cost Structure

Platform development and maintenance.

Marketplace updates and maintenance.
Storage of Backup.

Automation: Reduced cost to keep the platform running.

Revenue Streams

Fixed pricing. Subscription fee – customers purchases a Cloud App which functions for as long as they keep paying.

Indirect / None. Charged through End User. %'s

Fixed-value. A relatively large xxxxx kr. Sign-on fee is charged.
Indirect / No running revenue, charged through End User. %'s

www.businessmodelgeneration.com

Figure 12.: Cloud Apps business model [11].

B

MEEBOX BUSINESS MODEL INCLUDING CLOUD APPS

This business model focuses on the big picture and how Cloud Apps fits into Meebox's existing organization. It was introduced in the first report in this project [11].

The text marked with blue shows Meebox's existing organization. The text marked with red shows what Cloud Apps adds to Meebox's organization, and how this intrapreneurship can take advantage of the existing organization. The full explanation of the business model is found in [11] on page 62.

BM-Prototype 1

Entrepreneur-project ____
Intrapreneur-project _X___

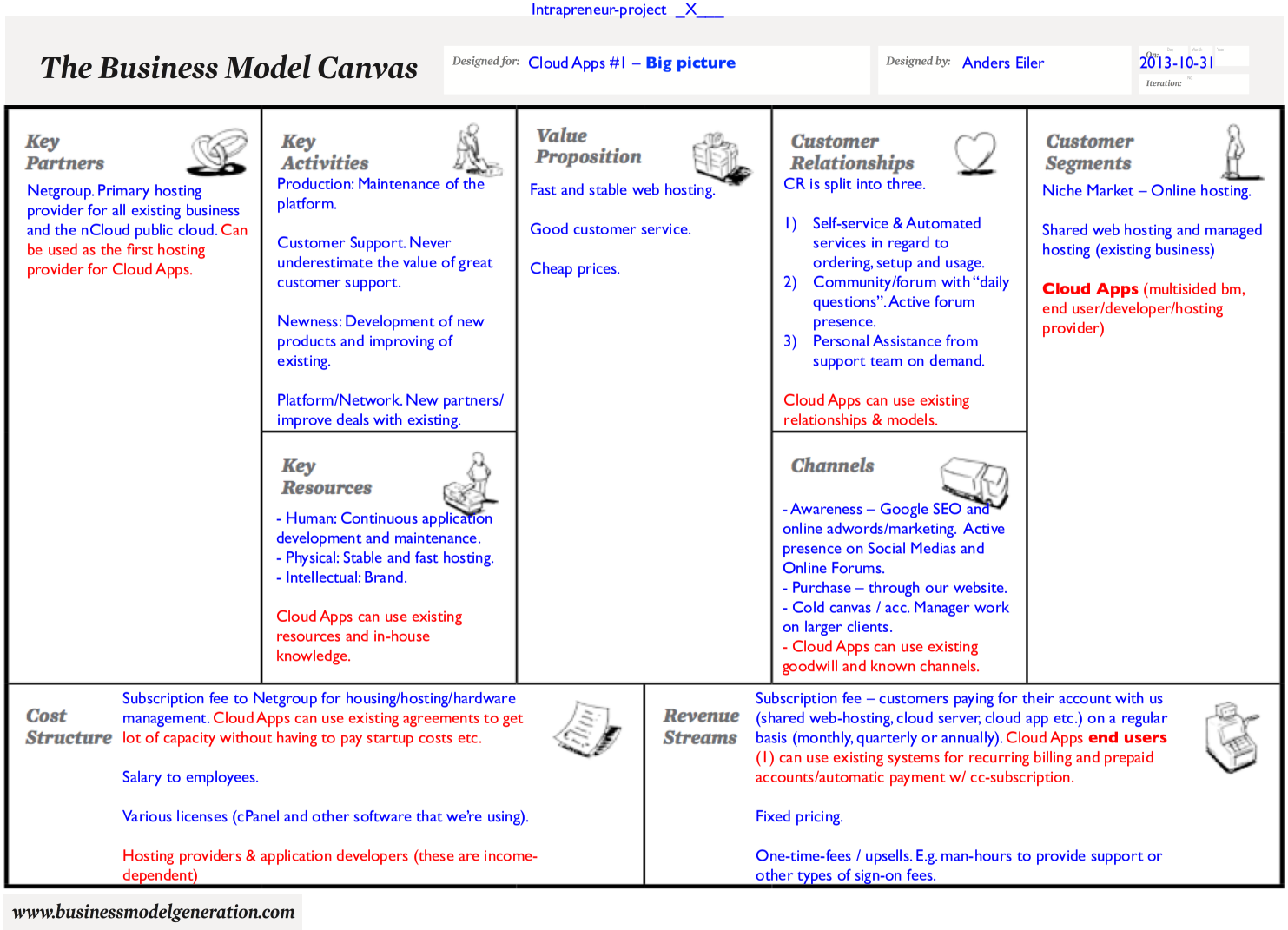


Figure 13.: Business Model for Cloud Apps showing how it fits into Meebox's existing organization. From [11]

C

CHECKLIST FOR IDEA EVALUATION FOR APPLICATION DEVELOPERS

The list below shows the Checklist for Idea Generation from Section 5.1. It is originally shows as a table, but due to the length when answers are included, it will be displayed as a bullet-list below.

1) Have you considered all the advantages or benefits of the idea? Is there a real need for it?

Allowing 3rd party Application Developers to create custom applications for the Cloud Apps marketplace will give three advantages: 1) It will allow for applications that would not otherwise be available. That will 2) Attract End Customers that may not have chosen to use Cloud Apps, which means that 3) both Application Developers, Hosting Providers and Meebox will make higher profits.

2) Have you pinpointed the exact problems or difficulties your idea is expected to solve?

By allowing Application Developers to create custom applications for the Cloud Apps marketplace, it will ensure that the marketplace always has a large number of relevant applications to offer to the End Customers. That will ensure that the platform does not risk to loose momentum and always stay on top.

3) Is your idea an original, new concept, or is it a new combination or adaptation?

The idea is not new. We are not sure who were the first to implement the concept of having a marketplace / store of applications that can be installed, but many large international companies have had success using the idea. It has been modified to fit into the Cloud Apps platform.

4) What immediate or short-range gains or results can be anticipated? Are the projected returns adequate? Are the risk factors acceptable?

There are no short-rage gains from implementing the idea, quite the opposite. The Application Developers will receive a free development environment in which they can create new applications. The risk and cost of making the development environment freely available to the Application Developers are acceptable because the long-term gains are expected to be much higher.

5) What long-range benefits can be anticipated?

Over time the Application Developers will accumulate a large number of applications that are only available as Cloud Apps. That will attract more End Customers due to the exclusivity.

6) Have you checked the idea for faults or limitations?

We were not able to find major flaws or limitations with this idea.

7) Are there any problems the idea might create? What are the changes involved?

Allowing Application Developers to create custom applications for the marketplace and making a development environment freely available for them requires an amount of resources from our side to create the facilities to do this. It requires some changes to the marketplace because it now has to support the 3rd party applications. At the same time we need to determine whether or not all 3rd party applications should be approved by a "board" to ensure the quality of the applications that are made available.

8) How simple or complex is going to be the idea's execution or implementation?

The concept is simple but we expect the implementation to be complex because there are many things that needs to be implemented.

9) Could you work out several variations of the idea? Could you offer alternative ideas?

In the pursuit if a community-like approach to application development, we believe that a marketplace where 3rd party developers can submit their own applications is the best approach.

10) Does your idea have a natural sales appeal? Is the market ready for it? Can customers afford it? Will they buy it? Is there a timing factor?

Other companies have already proved that a marketplace where 3rd party Application Developers can submit their own applications is working. Because the Application Developers can set the price/license fee themselves it is the marketplace itself and the End Customers who determine if an applications is too expensive. If it is, nobody will buy it thus the Application Developer should consider to lower the fee.

11) What, if anything, is your competition doing in this area? Can your company be competitive?

At this point, there are no competitors with the same approach.

12) Have you considered the possibility of user resistance or difficulties?

Because other companies have already implemented similar marketplaces with great success, we do not feat user resistance against the idea.

13) Does your idea fill a real need, or does the need have to be created through promotional and advertising efforts?

It is a combination. On one side, the marketplace creates a need because it offers applications that the End Customer perhaps did not know they needed. On the other side, it must be assumed that the End Customers comes to the marketplace with the intend of buying a Cloud App that fulfills some purpose or solves a task. Therefore it will answer a real need in that case.

14) How soon could the idea be put into operation?

The Cloud Apps marketplace can be launched together with the rest of the platform.

15) What impact will your idea have on the product/business?

The platform could work without Application Developers, but we believe that they will add another dimension to the platform. It will allow for applications that would not otherwise be available in the marketplace and could give a competitive edge when the End Customer has to choose a platform for their hosting.

DECLARATION

Aalborg, January 2015

Anders Eiler