Data- and Value-Driven Software Engineering with Deep Customer Insight

PROCEEDINGS OF THE SEMINAR NO. 58314308

JÜRGEN MÜNCH (ED.) 17.12.2014

Faculty of Science

Department of Computer Science

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

EDITORS

Jürgen Münch (ed.)

ABSTRACT

There is a need in many software-based companies to evolve their software development practices towards continuous integration and continuous deployment. This allows a company to frequently and rapidly integrate and deploy their work and in consequence also opens opportunities for getting feedback from customers on a regular basis. Ideally, this feedback is used to support design decisions early in the development process, e.g., to determine which features should be maintained over time and which features should be skipped. In more general terms, the entire R&D system of an organization should be in a state where it is able to respond and act quickly based in instant customer feedback and where actual deployment of software functionality is seen as a way of fast experimenting and testing what the customer needs.

Experimentation refers here to fast validation of a business model or more specifically validating a value hypothesis. Reaching such a state of continuous experimentation implies a lot of challenges for organizations. Selected challenges are how to develop the “right” software while developing software “right”, how to have an appropriate tool infrastructure in place, how to measure and evaluate customer value, what are appropriate feedback systems, how to improve the velocity of software development, how to increase the business hit rate with new products and features, how to integrate such experiments into the development process, how to link knowledge about value for users or customers to higher-level goals of an organization. These challenges are quite new for many software-based organizations and not sufficiently understood from a software engineering perspective.

These proceedings contain selected seminar papers of the student seminar Data- and Value-Driven Software Engineering with Deep Customer Insight that was held at the Department of Computer Science of the University of Helsinki. The seminar was held during the fall semester of 2014 from September 1st to December 8th. Papers in the seminar cover a wide range of topics related to the creation of value in software engineering. An interview of startups shows that emerging companies face a number of key decision points that shape their future. Value has a different meaning in different contexts. Embedded devices can be used to gather data and provide more value to the users through analysis and adaptation to circumstances. In entertainment, metrics can provide content creators the chance to react to user behavior and provide a more meaningful user experience. Value creation needs an active approach to software development from the companies: software engineering processes need to be incorporated with proper mechanisms to find the correct stakeholders, elicit requirements that provide the highest value and successfully implement the necessary changes with short development cycles. When the right building blocks are in place, companies are able to quickly deliver new software and leverage data from their products and services to continuously improve the perceived value of software.

KEYWORDS

value creation, Internet of Things, agile development, goal-oriented requirements, impact mapping, user stories, stakeholder analysis, DevOps

PAGES	**LANGUAGE**
49	English

Table of Contents

Startups Pivoting Towards Value

Kasper Hirvikoski
University of Helsinki
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2 B)
FI-00014 University of Helsinki
khirviko@cs.helsinki.fi

***Abstract*—Everything starts from an idea, but it is not always obvious what the value of that idea is. There comes a point in time — a turning point — when a company chooses to proceed a new path. These decisions form into key pivoting points, when a company changes their strategy on a product or service or even the whole company. An idea is evolved to a certain direction, without knowing what the outcome will be. Sometimes these pivots succeed, sometimes the new strategy ends up in a failure. These previous successes and failures can provide something essential that can help new startups to create their path and learn with an Agile and Lean mindset. Nowadays collecting valuable data about user behaviour has become more and more straightforward. This data can then be used as the basis for a pivot. However, it is not always about data. Intuition — or even pure luck — plays a significant role in many success stories. An entrepreneur has a feeling for the right direction. In many cases, the need for a product or service is created without a necessity. Users do not most of the time know what they need. Therefore, it is valuable to have a historical view on how startups have found their path to value by pivoting and what the process has been behind it.**

I. Introduction

Most startups start with a raw idea of a product or service. It is not uncommon that the idea will change quite a bit during its lifetime. Andy Whitlock, a product strategist, drew a fitting mental picture about this [1]. You see the road ahead as a clear and straight path to an objective you have set. What you do not always realise, is that the path will have its twists and turns along the way. What you can really only do, is to plan to a certain point ahead. The rest of your path will be a gloomy fog in the distance. You need to be ready to make difficult choices along the way. These choices form into pivots. A pivot is considered a point in time, when a strategy of a product or service is altered significantly.

Startups are created from a spur of the moment. A group of talented people realise a gab in the market, come up with an idea, and try to capitalise on that. Some have estimated that the failure rate for startups on the IT-sector is up to 90% [2]. The prevalent mindset is to: "fail fast, fail often." One in ten startups succeed. Others estimate that over a half of startups are shut down in a matter of years after their outset [3]. It seems most startups fail and many entrepreneurs overestimate the chance of success and what the costs of failing are [2]. That does not mean that failing would not be a path to success. It can lead to something successful.

The amount of startups has increased at a rapid pace. In 2012, the number of seed investment deals had tripled in the US, from three years earlier [2]. Startups are seen as a new way to do creative things. In fact, many startups are being sold for millions without a business model [4]. Larger companies acquire startups to learn new things, fill their product lines and to recruit talented people.

Continuous innovation, the ability to renew organisational structures and develop new ideas and products, has been widely acknowledged as a key issue for long-term progress. However, it has proved extremely hard to accomplish in practise [5]. History is full of innovative companies that have struggled to make changes when needed. They have lost their ability to innovate and ceased to exist.

Choices can be backed by data. Collecting data about how the users use your products or services has become less strenuous. With the likes of continuous integration, continuous delivery and continuous experimentation, startups can develop products in fast iterations, with always keeping the features up to date [6]–[8]. Especially on the web, new versions of products can be deployed without the need of user interaction [6], [9]. This allows startups to experiment and explore new paths easily. With prototyping, A/B testing and other controlled experiments, startups can collect data about the causality between changes and their influence on observed user behaviour [6], [9]. This can be used to determine what is the value between different implementations of a feature.

Sometimes data is not everything. Intuition — or even pure luck — plays a significant role in many success stories. An innovative and great entrepreneur has a notable influence on a success story [10]. In many cases, the need for a product or service is created without a necessity. That is, users do not always know what they are looking for a given problem or even if the problem exists.

Most pivots happen at a moment of desperation. An idea is not working or the users are not immersed by it. These moments can create a realisation — an "aha"-moment — that can make or break a startup. Shifts in market and technologies force startups to reinvent ideas at a growing pace.

It is therefore valuable to have a historical view on how startups have found their path to value by pivoting, and how the process has evolved.

II. Historical Background

There are "classical" examples on how companies have grown from something else to what we know them as now.

Nokia has a long history of successful changes and innovation. From its humble beginning in 1865 with one paper mill,

it has worked among other things on cables, tires, rubber boots and more lately mobile phones and networks [11]. Recently, Nokia has been in the spotlight by struggling with the shift in the mobile phone market.

More current examples are todays leading social networks Facebook and Twitter.

Mark Zuckerberg launched Facebook as a Harvard sophomore in 2004 [12]. He had previously developed a controversial platform for ranking Harvard students by their attractiveness. Even though contentious, it had become a success at the campus.

It has been stated that Zuckerberg used an idea originally created by three seniors Cameron Winklevoss, Tyler Winklevoss and Divya Narendra [12]. They had an idea of creating a social network for Harvard students and alumni. Zuckerberg was employed as a replacement to continue the development work for the network. Instead of building the platform pitched by the seniors, Zuckerberg allegedly delayed the project and decided that he had an improved idea [12]. He build a competing platform and friendships were harmed.

Nevertheless, within 24 hours, 1200 Harvard students had signed up for Facebook. After one month, over half of the undergraduate population had a profile [13]. The network expanded promptly to other universities. Eventually in 2006, Facebook was extended beyond its educational root and it became available to everyone. What was originally aimed to a very focused user base, had expanded to a global phenomenon.

Twitter started from the ruins of a podcasting platform Odeo [14]. Noah Glass had invented a product where a message dictated to a phone number would be turned into a MP3-audio file hosted on the Internet. This became the core for Evan Williams's startup. Williams who had previously worked for Google, asked a friend, also a ex-Googler, named Biz Stone to join the startup. At the same time, Jack Dorsey worked as web-designer for Odeo.

In 2005, Apple released a podcasting service for iTunes. This eventually made Odeo obsolete. A product was built, but never used. They were not emotionally invested in the product anymore [14].

Odeo started holding "hackathons" to make employees come up with new ideas. Employees spent a whole day working on new projects in different groups. Glass, who had admired Dorsey, joined with him. Dorsey had an idea of creating a product that revolved around "status" — what people were doing at a given time. They came up with an idea with fellow group member Florian Weber. The idea was to send a text message to one number and it would then be broadcasted out to all of your friends: Twttr. Williams was skeptical, but Glass ended up leading the project that eventually ended up as Twitter.

In August 2006, a small earthquake shook San Fransisco and word quickly spread through Twitter. This was the first "aha"-moment. Different minds had joined together to create a product from the ruins of another. The road was not simple. Investors were not convinced and Williams ended up buying Odeo's stock. Friendships were broken. Twitter had 5000 registered users and the future was a mystery. The rest is history, they made a risk and it payed off. Pivots require courage [8].

In more "modern" examples, GitHub was born on an emergent technology that was not widely used [15]. A friend and former coworker Dave Fayram, introduced Tom Preston-Werner in a new version control system called Git. Git was supposed to be an easy way to work on code in a distributed way. The Ruby community was quickly adopting Git, but there was no commercial hosting available for sharing Git-repositories easily, securely and privately. Other solutions were not ideal and required manual work. GitHub was then born in 2007. They were creating a service in a market that did not exist [15]. It would take time, but the platform was ready when the users started embracing Git.

The Internet has made it possible for many companies and their business models to exists: Facebook, Twitter, GitHub and so on. In fact, the Internet has made it possible to also work remotely. Many startups work this way at the beginning. Teams work remotely, but get together for planning and discussion [15], [16]. This is how GitHub worked in its early days and how Travis CI — a startup focusing in building a continuous integration platform — works today.

Kickstarter's idea was launched in 2002, when Perry Chen hoped of having an artist play at a Jazz-festival in New Orleans [17]. The show did not happen, but it gave him an idea of starting a platform where people could crowdfund ideas if they would hit a set funding goal. In 2009, in the bloom of the Internet startup-era, Kickstarter was launched to the public. From there on, it has grown to be one of the leading platforms for getting funding for your ideas.

There are strikingly similar characteristics behind these examples. In Facebook's and Twitter's cases, the pivot has formed on a strategical change. For Facebook, the user base has changed, and for Twitter a single feature has formed the base for another product [12], [14]. This is key for their successes. GitHub, Travis CI and Kickstarter have built products to a market that did not exist before [15]–[17]. This is not uncommon upon startups.

III. Related Work

Ries has written a comprehensive book about the essence of a Lean startup [8]. This paper has been structured to evaluate the ideas and observations presented by the author.

Moreover, not much scientific research has been done on the subject of how startups pivot to value. Most of the published work has focused on empirical analysis of case studies. The literature focused on the subject has been written outside the research field.

Steiber and Alänge have conducted an interview based empirical study on how continuous innovation characterises at Google [5]. They found out that a dynamic and open corporate is essential in invoking innovations in regular work. This is established by orienting towards innovation and accepting change. A innovation-oriented board, management, culture, competent and committed individuals and leaders have a strong impact. Leaders who empower, coach and remove obstacles to innovation make this possible, but individuals need to be essentially trusted.

Trimi and Berbegal-Mirabent researched how business models affect the competitive edge of startups [18]. Business models outline broader plans to create value than purely the core strategy of the company. Startups working in a technology-intensive sector may face constraints such as large investments required to develop a product, very short product life cycles and the emergence of competitors. The emerging trend in business model design highlights the importance of not only listening to customers, but also creating new values or products with customers. It is key that businesses make faster choices [18].

Davila et al. have researched how venture capital financing (VC) impacts the growth of startup firms [19]. They conclude that VC funding events signal about the quality of a startup. Their findings suggest that the lack of timely funds can delay the growth of a startup. They also suggest that the growth in employees, compared to equity valuation, can be seen as a better proxy for changes in the value of a startup.

IV. Research Method

My research method for this paper was reviewing the current industry, the new startups and established, and creating a historical look on pivoting.

I also conducted several interviews with startups and entrepreneurs behind them, to get their views and story behind making changes in their strategy and what came out of this [20], [21]. My research is a qualitative research based on these interviews and topics presented by them.

Koen Bok is an entrepreneur and angel investor based in Amsterdam [20], [22]. He founded Sofa, an indie startup, building software and interaction design. Sofa was focused on developing successful products for the Mac-platform and was then acquired by Facebook in 2011 [23]. He (and part his team) then worked for Facebook for two years, after he returned back to Amsterdam and started a new startup called Podium. Podium focuses on building web-development and prototyping tools.

Kyle Bragger is an entrepreneur based in Berlin, who founded and sold Forrst, an online community for designers and developers [21], [24]. He then has joined to work on products at Elepath. He is the co-founder and lead-engineer of a startup called Exposure, a web-platform for creating and posting photo stories established in 2013.

In addition, the structure of this paper is backed up by conducting a literately review about the subject from research papers to blogs.

V. Approach

The research question is: what could we learn from these previous pivots going ahead. What was the motivation behind pivoting and did the startups have indicators that where considered useful for making these decisions?

My aim was also to look at these findings based on Ries's ten most common types of pivots and trying to find common factors behind them [8]. The objective was to evaluate what has been the data and value behind pivoting.

I conducted an interview with startups with the following questions:

A. Background

1) What was the main reason why your company was started?
2) What was the main obstacle when you started?
3) When did you face this obstacle?
4) How did it affect your plan for the product or company?

B. During Your Journey

1) Have you changed your product or company strategy radically? When did it happen?
2) What was the main reason for this decision?
3) Was the change motivated by intuition or some other factor, e.g. external indicators or data?
4) Did you have data available to analyse options for reacting to these findings?
5) When reflecting on the actions that followed, what was the main learning for the change of strategy?

C. Current Situation

1) How does your company evaluate how the user values your products?
2) Do you collect automated data on how the user uses your products? If so, what?
3) Do you analyse that data to make changes?
4) What are the top three metrics that matter?

VI. Ries's Categorisation of a Pivot

A pivot is considered a point in time, when a strategy of a product or service — or even the the strategy of the whole company — is changed drastically. A new direction will be carried on from that point onward.

Ries has categorised ten most common types of pivots in his book *The Lean Startup* [8]. These are the following:

1) **Zoom-in Pivot.** A single feature in a product becomes the whole product.
2) **Zoom-out Pivot.** The reverse, where a single feature is insufficient to support the whole product. In this case, the whole product becomes a single feature of a larger product.
3) **Customer Segment Pivot.** The product solves a real problem for real customers, but for a different customer base than originally anticipated. The product hypothesis is partially confirmed by solving a right problem. The product needs to be optimised.
4) **Customer Need Pivot.** Customer feedback indicates that the problem solved is not very important. However, because of the customers, a new related problem might be discovered. In many cases, a completely new product must be developed for this need. The hypothesis is again partially confirmed, the target customers have a problem worth solving.
5) **Platform Pivot.** A change from an application to a platform or vice versa. Most commonly, startups create a single application, a so-called killer app, for their platform. Only later, does the platform emerge as a way to support third parties creating their own

related products. This can also happen the other way around.

6) **Business Architecture Pivot.** You can choose to operate your business architecture as high margin and low volume (niche market), or low margin and high volume (mass market). According to Geoffrey Moore, you cannot do both at the same time [8].
7) **Value Capture Pivot.** A change in the way the value created by the company is captured. Usually referred to how the company creates revenue from the product. Changes is monetisation can have far-reaching effects.
8) **Engine of Growth Pivot.** A change in the growth engine: the viral (word of mouth), sticky (retaining customers) and paid (buying customers) growth models. The right model can affect the speed and profitability of growth.
9) **Channel Pivot.** Changing the way the product is delivered to the customer. A different channel can have greater effectiveness.
10) **Technology Pivot.** Discovering a completely different technology to implement the same solution. The new technology can provide a better price or performance compared to the existing one.

VII. Making Pivots

Is is difficult to choose when to pivot. These decisions can be backed by anything from intuition to external indicators such as user feedback. In any case, making changes requires courage and determination [8]. There are key lessons to take into account for pivoting. The following sections are a combination of my thoughts and the thoughts I have gathered.

A. Start Early

An entrepreneur with a big vision and stubborn determination can charge through obstacles and make whatever their ambition is. The passion, energy and vision that people can bring to new ventures are resources that should not be disregarded [8].

At the same time, the uncertainty of a product usually requires many course correction or pivots to find success. Therefor, it is helpful to start by creating a product with the smallest set of features and move that to the market as quickly as possible (minimum viable product) [8], [15]. This makes it possible to test reactions, learn and iterate. This is often called as the build-measure-learn cycle or continuous innovation [8]. See figure 1. Through this process, a startup can learn if and when to make a sharp turn, a pivot. This methodology can also be used in making decisions about the business model of the whole company [18].

Often, the first one to the mass market will eventually win. Creating a product early on emergent technology, provides a platform to build a product with basically no overhead, no competition and without time pressures [15].

B. Learn and Adapt

A startup can only afford to make a certain amount of pivots [8]. In most cases, you need to either achieve lift-off or fail. Especially when there are investors trying to secure their investments back. Sometimes it is just too late to make a pivot. The build-measure-learn cycle and other Lean-methods sustain innovation. This is key for long-term economic growth [8].

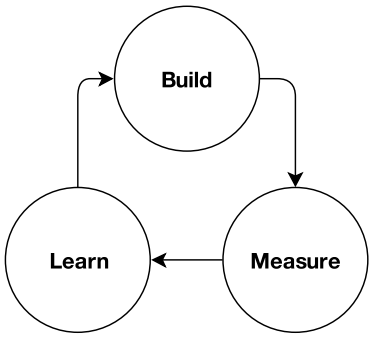

Fig. 1. Build-Measure-Learn Cycle

The fundamental activity of a startup is to turn ideas into products, measure how the user reacts to these, and then learn whether to pivot or to keep going. Innovation and speed are key premises [18].

As startups come to understand their customers better, they are able to improve the product. The data behind how the customer interacts with the product is indispensable [6], [8]. According to Preston-Werner, one of the founders of GitHub, creating a new startup is an intense learning experience [15]. It is a learning experience through mistakes and achievements.

You need to listen and pay attention to your customers, but with such care, that you do not let them tell you what to do [15]. In most cases, users do not know what they want, until you give them what they want. Users will have generic problems, but they will perceive them as very specific ones. A savvy entrepreneur needs to see the bigger picture, the underlying question, behind these problems. This is where a functional and elegant solution can be found. The more everyone talks to the customers, the more beneficial it can be in determining whether you are going in the right direction [16].

C. Make Changes

With continuous integration, continuous delivery and continuous experimentation, startups can develop products in fast iterations with always keeping the features up to date [6]–[8]. Especially on the web, new versions of products can be deployed without the need of user interaction [6], [9].

In the early days of GitHub, new features where deployed up to ten times in one afternoon [15]. As a matter of fact, Amazon deploys new software every 11,6 seconds [25].

Users will forgive a small amount of downtime, if the features will improve the product. You need to get most out of your first years though, because if and when a product will become more popular, you will need to be more careful about these choices [15]. Once established, downtime and defects will have a larger role in affecting the reputation of a startup.

D. Evaluate the Data

Making decisions purely based on intuition can be risky. Learning, adapting and making changes are guided by data [8].

Data-driven decisions are good to proof you are on a wrong path, but not necessary for creating new and innovative ideas [20].

In most cases even intuition is guided at least subconsciously by data [20]. You realise a solution is not working the way you intend it to work and therefor you see a new path to pursue. Tracking key metrics, from simple user interactions, such as clicks, to revenue or user retention (users sticking around for a long time), can provide guides for evaluating your path.

E. Have Trust in Your Product

One key lesson is to have trust in your product. At start it may be tricky to convince people of your product [14], [20]. Getting positive feedback and support is essential [20], [21]. Making changes is valuable, but you also need to know when to stick with a decision. Users will not adapt new solutions immediately.

A startup needs to be concise with its strategy. Many startups fall in the trap of figuring how to create profit with their idea. Choosing the right growth engine: viral, sticky or paid is important [8]. It can be hard to make a change from a free-model to charging for a service. Make the choice early on. Users are willing to pay for good services. Also, building a startup with the intentions to sell to a bigger company, may result in users being left stranded.

F. Need for Investors

You do not always need funding [15]. Some startups try to grow faster than necessary. Investors bring you capital, but also restrict how you proceed with your plan. It is worth to grow your startup with care and taking your time.

You will eventually prove your value with your product [15]. Nevertheless, some findings suggest that the lack of timely funds can delay the growth of a startup [19]. For first-time entrepreneurs, there is always some financial risk involved [20]. With a good track record, this will ease in future ventures.

G. Have Fun

A dynamic and open corporate is essential in invoking innovations in regular work. A innovation-oriented board, management, culture, competent and committed individuals and leaders have a strong impact. Leaders who empower, coach and remove obstacles to innovation make this possible, but individuals need to be essentially trusted [5].

According to Preston-Werner, working on a startup is a challenging project, but most of the time you are working with your best friends and having a great time [15]. Maintaining a playful working environment, will have a positive effect on the workers health, as well as the startups health.

Personal life comes first [16]. Giving time to disconnect, will make sure work is done at a normal pace. A happy team will most likely have happy customers [16].

VIII. Interview Results

Based on the interviews, I have gathered the following about facing obstacles and making pivots.

A. Facing Obstacles at Start

One of the main obstacles is to choose what to work on [20]. Koen Bok says, you want to pick a problem that a) has a chance on success, or can sustain itself and b) is fun to work on for a long time.

Apart from those two important ones, Bok feels that it is also important, if you can easily explain anyone what you are doing. It is also nice if people you respect can see and use your work [20].

For first-time entrepreneurs, there is also some social pressure and financial risk involved. He thinks that it lessens with every startup you do [20].

Kyle Bragger feels the main obstacles when starting are convincing folks that a product is better and not like other products out in the market [21]. These obstacles are ongoing and will never go away.

He says, it is so important to understand why your product exists and to build for a specific audience [21]. You need to always be focusing on clearly communicating what problem you are solving for people.

B. Pivoting

Bok does not see pivoting as a radical change [20]. A startup does that every day in the beginning, but because everything is still so small it is hard to see the “radicalness”. He feels that you know that these small changes have huge implications in the future: whether your product will get used and succeed or not.

Bragger states that they have not needed to change their strategy radically at Exposure [21]. However, maintaining user retention is an ongoing project that requires focus.

C. Evaluate the Data

Whether to evaluate data or trust your intuition, Bok feels you need ideally a combination of both [20]. In his experience, data-driven decisions are good to proof you are on a wrong path. They also help to make an optimisation in a (part of) your product. Bok however states, that they never really help you invent things in a creative way, which is how big breakthroughs materialise [20].

On the early days of a startup, data can be pretty misleading, because the numbers are so small [20]. Bok says, your conclusions cannot really be significant, but they somehow feel supported by the data. The data influences how you make choices, because that is how you get to the conclusions.

At Exposure, on the automated side, they measure traffic analytics like most web-applications [21]. Bragger also describes that they also measure a lot of metrics specific to their product. These include for example how many photos someone might upload per post and on average, how long users work on a post before publishing. They also use their intuition based on user feedback, support requests, and so forth to evaluate how specific features are working.

D. Key Metrics

Bok says the key metrics that he pays attention to are: total active usage per a variable, total revenue and user retention over time [20]. Bragger says their key metrics are monthly recurring revenue, engagement (how many people are using the product e.g. daily), and retention [21].

IX. Evaluation

It seems clear that social factors play a big role in successes and failures. In the case of Twitter, the right minds joined together to work on an idea [14]. This was a key factor in the beginning of GitHub and Kickstarter as well [15], [17]. However, a social pivot seems to be missing from Ries's categorisation of the ten most common types of pivots altogether [8].

Startups — or even established companies — can make or break on people. A mixture of right people can create a constructive environment for creativity and innovation. Moreover, a wrong person, in a wrong position, can have a negative effect on a startup's path. This is true for established businesses also [10].

Therefor, I would suggest an addition of a *Social Pivot*, where active changes in social factors, such as persons and environments, change the direction of a company.

It also seems evident, that one can make enemies along the way. In both Facebook's and Twitter's case, friendships were harmed during the course of the startup [12], [14]. People with same aspirations can become powerful allies or dangerous foes. It is easier to work with people that have shown relevant past performance than to project the future [15]. It was only after Steve Jobs return to Apple, whereof Apple grew to the success we know it as today [10]. The road was not smooth for him or Apple.

You need to trust your team to embody the startup's vision. Sharing a vision can help motivate the employees in moving a product forward [16]. This is a founding concept in Agile development practises. Micromanagement will not work, instead you need to sustain a culture where teams can move and innovate with the experimentation system [8].

Ries's categorisation of pivots can be seen as a good starting point for evaluating changes [8]. The most obvious pivots are those that include zooming in or zooming out. In Twitter's case a single feature became the starting point for a completely new product (Zoom-in Pivot) [14]. For Facebook, the customer segment changed from an internal platform to a global phenomenon (Customer Segment Pivot) [12]. The other pivots can also have outer facing effects, but can also be more internal changes in the strategy of the product or its market.

X. Limitations

It is worth to note that case studies tend to be tightly related to their contexts. Making objectional and universal conclusions from their findings is difficult. Each setting is unique and cannot directly be compared to another. They do however give insightful information about the process behind.

Starting a startup can be a leap to the unknown. These stories have differences, but in their essence they share many similar characteristics, which can give some thoughtful ideas and opinions to be considered.

The opinions and ideas in this article are as referenced. They might not have a universal truth and should be treated as such. What works with one startup, might not work with another.

XI. Conclusions

No startup is the same. No path to success or failure is the same. A startup's work is never done [8]. The trial-error philosophy for validating hypotheses boosts innovation and encourages the creation of products in a much faster time span. This helps entrepreneurs to start ventures with a stronger assurance of success [18].

For startups, pivoting does not appear as drastic changes in the strategy of a product. In effect, pivoting for startups is daily decisions that lead to success or failure. Only later, with reflection, can we see key moments that have influenced in that path. For many entrepreneurs their whole career consists of pivoting. One failure leads to a success. Fail often, fail fast to learn something new. Have the courage to try.

A startup can only afford to make a certain amount of pivots [8]. In most cases, you need to either achieve lift-off or fail. One of the main obstacles is to choose what to work on [20]. Koen Bok says, you want to pick a problem that a) has a chance on success, or can sustain itself and b) is fun to work on for a long time.

Making decisions purely based on intuition can be risky. Learning, adapting and making changes are guided by data [8]. Data-driven decisions are good to proof you are on a wrong path, but not necessary for creating new and innovative ideas [20]. The data influences how you make choices, because that is how you get to the conclusions.

Social factors have a significant role in pivots. Whether these are caused by changes in persons or environments. People make or break ventures. Most successes are a combination, a mixture of talented people. This is the last factor that makes an idea work. Ries's categorisation of ten most common types of pivots seems to lack a *Social Pivot* [8].

An entrepreneur with a big vision and stubborn determination can charge through obstacles and make whatever their ambition is. The passion, energy and vision that people can bring to new ventures are resources that should not be disregarded [8].

The path to success is more easily achieved by evolving gradually. Start early with a concept, build a minimum viable product, observe, learn and adapt.

XII. Summary

Everything starts from an idea, but it is not always obvious what the value of that idea is. There comes a point in time — a turning point — when a company chooses to proceed a new path. These decisions form into key pivoting points, when a company changes their strategy on a product or service or even the whole company.

Nowadays collecting valuable data about user behaviour has become more and more straightforward. This data can be

used to make decisions. Data is not everything. Sometimes it can only tell when you are doing something wrong, but it cannot tell which direction you should take.

It is interesting to know what we could learn from these previous pivots going ahead. What was the motivation behind pivoting and whether the startups have had indicators that where considered useful for making these decisions.

Each path is unique, but there are clearly best practises that can be followed. Start early, evolve, adapt and learn. The three main steps are to build, learn and measure. A startup will pivot daily. Work is never done. Small decisions clarify the vision. The rest is what makes you unique.

XIII. Future Work

My main intention is to incorporate part of these ideas and findings into my Master's thesis. In the future, it would be interesting to create a larger case study on this subject matter with a broader perspective.

XIV. Acknowledgements

I would like to express my gratitude to Hanna Mäenpää and Arto Vihavainen for reviewing and guiding the research method for this paper.

References

[1] A. Whitlock, Twitter, 2014. [Online]. Available: https://twitter.com/andywhitlock/status/524545897737494528/

[2] R. Carroll, "Silicon Valley's culture of failure ... and "the walking dead" it leaves behind," The Guardian, 2014. [Online]. Available: http://theguardian.com/technology/2014/jun/28/silicon-valley-startup-failure-culture-success-myth/

[3] J. Saarinen, "Yli puolet startupeista ei lähde lentoon: "Totta kai se hävetti"," Helsingin Sanomat, 2014. [Online]. Available: http://hs.fi/talous/a14149011111091/

[4] D. Aujla, "Why startups sell for millions with no business model," Inc., 2014. [Online]. Available: http://inc.com/dev-aujla/why-startups-sell-for-millions-with-no-business-model.html

[5] A. Steiber and S. Alänge, "A corporate system for continuous innovation: the case of Google Inc." *European Journal of Innovation Management*, vol. 16, no. 2, pp. 243–264, 2013. [Online]. Available: http://emeraldinsight.com/doi/abs/10.1108/14601061311324566

[6] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data Mining and Knowledge Discovery*, vol. 18, no. 1, pp. 140–181, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10618-008-0114-1

[7] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, ser. The Addison-Wesley Signature Series (Fowler). Pearson Education, 2010.

[8] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011.

[9] K. Hirvikoski, "Streamlining A/B testing on the web and its relevance to embedded systems," in *Proceedings of the Seminar No. 58314109, Real-Time Value Delivery in Software Engineering*. University of Helsinki, Department of Computer Science, 2014, pp. 7–13. [Online]. Available: https://www.cs.helsinki.fi/intranet/group/smqm/rtvdse14/rtvdse14_proceedings_public.pdf

[10] W. Isaacson, *Steve Jobs*. Simon & Schuster, 2011.

[11] Nokia, "Our story," 2014. [Online]. Available: http://company.nokia.com/en/about-us/our-company/our-story/

[12] N. Carlson, "At last — the full story of how Facebook was founded," Business Insider, 2010. [Online]. Available: http://businessinsider.com/how-facebook-was-founded-2010-3/

[13] S. Phillips, "A brief history of Facebook," The Guardian, 2007. [Online]. Available: http://theguardian.com/technology/2007/jul/25/media.newmedia/

[14] N. Carlson, "The real history of Twitter," Business Insider, 2011. [Online]. Available: http://businessinsider.com/how-twitter-was-founded-2011-4/

[15] T. Preston-Werner, "Ten lessons from GitHub's first year," 2008/2011. [Online]. Available: http://tom.preston-werner.com/2011/03/29/ten-lessons-from-githubs-first-year.html

[16] M. Meyer, "How we roll as a team: Our company's work values," 2014. [Online]. Available: http://blog.travis-ci.com/2014-02-13-how-we-roll-as-a-team/

[17] Kickstarter, "A brief history of Kickstarter," 2014. [Online]. Available: https://kickstarter.com/stories/fiveyears/

[18] S. Trimi and J. Berbegal-Mirabent, "Business model innovation in entrepreneurship," *International Entrepreneurship and Management Journal*, vol. 8, no. 4, pp. 449–465, 2012. [Online]. Available: http://link.springer.com/article/10.1007/s11365-012-0234-3

[19] A. Davila, G. Foster, and M. Gupta, "Venture capital financing and the growth of startup firms," *Journal of Business Venturing*, vol. 18, no. 6, pp. 689–708, 2003. [Online]. Available: http://sciencedirect.com/science/article/pii/S0883902602001271/

[20] K. Bok, personal correspondence, 2014.

[21] K. Bragger, personal correspondence, 2014.

[22] K. Bok, "Koen Bok," 2014. [Online]. Available: http://koenbok.com

[23] Sofa, "We were Sofa," 2014. [Online]. Available: http://madebysofa.com

[24] K. Bragger, "Kyle Bragger," 2014. [Online]. Available: http://kylewritescode.com

[25] O'Reilly, "Velocity 2011: Jon Jenkins, "velocity culture"," YouTube, 2011. [Online]. Available: https://youtube.com/watch?v=dxk8b9rSKOo

Business Models for value generation in the Internet of Things

Christian Blythe
Department of Computer Science
University of Helsinki
Helsinki, Finland
Email: christian.blythe@helsinki.fi

***Abstract*—With the imminent explosion of the Internet of Things and the expected proliferation of smart connected devices in our everyday lives business will have to change how they approach value generation in this modern age of big data. The traditional product centred business models are limited and restrictive when applied to this new landscape and a new mindset revolving around information and services as the key value propositions is required. This paper examines the value of information and how the laws of information relate to the Internet of things paradigm. We then examine how those principles can be used to propose new business models that shift a companies focus from *value creation*, a traditional product based approach, to a *value capture* approach that focuses on value created by information. To demonstrate how companies can employ these value capture principles the example of Nest Labs is considered with a contrast of the traditional product based business model and a new information centered business model approach.**

Keywords—Internet of Things, business models, value creation, Nest Learning Thermostat

I. INTRODUCTION

The Internet of Things (IoT) is the phenomenon of digital technology being incorporated into previously non-digital products such as clothes, everyday household appliances and food packaging. These intelligent devices form a network of sensors and actuators which are connected to the Internet providing a huge growth in access to contextual, real time data and feedback control [1].

Currently businesses have incorporated IoT technology into a number of areas, such as supply chain management, industrial process control and environmental monitoring; however many of these developments have been limited to cost reductions and improvements in operating efficiencies of current business processes rather than realising connected smart systems and producing new service values [2]

We are now on the cusp of a new revolution of interconnected devices in our homes, in the products we buy and on our persons. The main drivers for this consist of device miniaturisation, reduced production costs, lower power consumption and pervasive connectivity through Wi-Fi, 4G-LTE and radio frequency identification (RFID) technologies such as near-field communication (NFC) [3], [4].

Estimates on the number of devices in the future vary, but range between 26 billion [5] and 50 billion [6] connected devices by 2020. Not only are we heading for an explosion of these internet nodes embedded in things in our everyday lives, there are high expectations of the benefits that IoT will provide to people in the future [7], [8]. There are a wide range of domains and applications where IoT has the potential to have a high impact on every-day life (Figure 1), with these new business opportunities providing huge potential for business revenue creation.

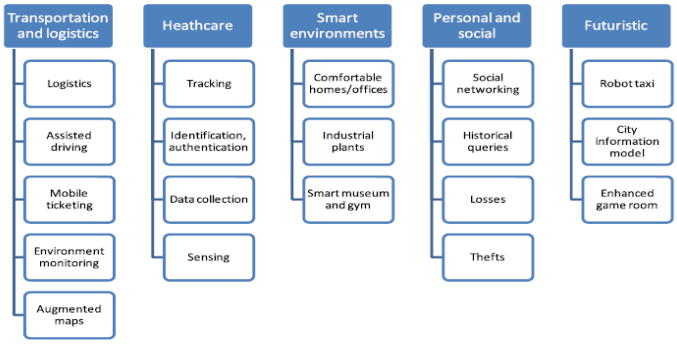

Fig. 1. Application domains and major scenarios [9]

For companies to be able to engage with, and benefit from these new technologies they may well need to alter their approach to what their products or services are and how their businesses are structured and operate. With many businesses (both new and established) employing business modelling to define their offerings, relationships with customers and the business structure, they are an obvious tool to consider how organisations might align themselves with an IoT marketplace.

With IoT introducing new volumes of data, at higher levels of granularity and with new potentials for integration and analysis it is arguable that data, and the information derived from it, should take a central role for companies operating in this field. In order to integrate IoT dynamics into business models it is advantageous to start examining information as a key component with its own potential for value generation.

This paper proceeds to outline the need for change within businesses approaches to IoT and how information can be used in business modelling as a core value proposition around which

the company can be structured to align it to an IoT marketplace. Section III examines information, its value and how that can translate to value generation for businesses. Section IV then reviews business models and describes how placing information as the central value proposition is the key aspect to alter business modelling to align with IoT. Finally the Nest Thermostat is used as an example to highlight the difference between the two approaches (traditional product centric and IoT centric) and how business models can be used to describe those differences.

II. BACKGROUND

The shift in the landscape of ICT has wide implications for businesses and Hui [10] argues that the use of current business frameworks and the streamlining of established business models will not be enough in the future. The traditional business model revolves around the product providing main value proposition [11] through the sale of the item, post-sale maintenance and sales of consumables (such ink cartridges for printers). These 'old style' companies face the challenge not only of converting their products into smart connected ones, but moreover of integrating them into a service based business model which has new and very different value propositions.

The change required for this cloud based environment involves a shift in mindset between traditional value creation to a value capture approach [10]. *Value creation* for traditional companies means identifying customer needs, engineering solutions for those needs and selling a product with the relevant features. This leads to feature-vs-feature competition, which when exhausted devolves into price wars reducing profitability until products become obsolete. The rush is then on to develop and sell the next product to maintain a business, often employing a re-examination of existing customer needs in a reactive manner.

		TRADITIONAL PRODUCT MINDSET	INTERNET OF THINGS MINDSET
VALUE CREATION	Customer needs	Solve for existing needs and lifestyle in a reactive manner	Address real-time and emergent needs in a predictive manner
	Offering	Stand alone product that becomes obsolete over time	Product refreshes through over-the-air updates and has synergy value
	Role of data	Single point data is used for future product requirements	Information convergence creates the experience for current products and enables services
VALUE CAPTURE	Path to profit	Sell the next product or device	Enable recurring revenue
	Control points	Potentially includes commodity advantages, IP ownership, & brand	Adds personalization and context; network effects between products
	Capability development	Leverage core competencies, existing resources & processes	Understand how other ecosystem partners make money

SOURCE SMART DESIGN HBR.ORG

Fig. 2. Traditional and IoT business approaches [10]

Value capture however incorporates the prospect of other revenue streams not limited the sale of physical products. These alternative revenue streams include the sale of value-added services, subscriptions and apps to consumers, but also can include the provision of information services to third parties and multiple collaborators. Figure 2 demonstrates the key differences between these two mindsets and demonstrates the need for the development of business models focusing on data and service value propositions rather then physical product sales.

With this new mindset comes the realisation of the importance of the role of information in creating value propositions. Not only the collection and use of data within a single company or application scenario, but the convergence of multiple, disparate data streams employed to create new meaning and value which are shared and used by multiple collaborators [2]. It is from this foundation that new business models can be developed for use within the interconnected world of the Internet of things.

III. VALUE GENERATION FOR IOT

Traditionally the money stream for businesses is exclusively linked to the product stream pricing, and any information generated from products or business activities tends to not have a specific value attached to it [11]. This tends to mean costs of information are hidden in the product pricing, leading to a mindset of reluctance to value, or consider paying for, information streams.

In the IoT information takes centre stage as a commodity with intrinsic value, even though every information stream is linked in some way to a physical device. It is therefore important to be able to view information as an asset in its own right rather than just a by-product of data capture from suitably enabled devices.

A. Laws of Information

Although it is clear how data and the information it yields is of core importance to IoT businesses, it can be difficult quantify how much of an asset it is. Whilst it is possible to measure the (increasing) costs of organisational resources spent on data capture, storage, processing and maintenance; it can be difficult to assess the value of information to the organisation.

In order to be able to quantify the value of information, to then be able identify potential value capture opportunities, Bucherer & Uckelman [11] cite Moody & Walsh's seven 'Laws of Information' [12] and explain their relevance to IoT as follows:

First Law of Information: Information is (infinitely) shareable and can be shared with Others without loss of value – Here the IoT provides access to data streams from products which can be shared any number of times with multiple collaborators. This data can be monetised through paid for access, and when the income outweighs the cost of data acquisition it becomes a viable value proposition (which is also scales with the number of 'information customers')

Second Law of Information: The value of information increases with use and it does not provide any value it is not used at all – The major cost factors related to data are collection, storage and maintenance of said data, with the cost of accessing being relatively small. The IoT provides a framework to not only reduce the costs of data acquisition, but also provides increased opportunities for that data to be used. One challenge is how to ensure your data is known about by potential consumers and that it is useful to them so it does get used (and therefore have its value increase)

Third Law of Information: Information is perishable and depreciates over time – The IoT provides the capacity for real-time data gathering to provide the benefits of contextual information and life cycle information access. In addition to this the growing amount of historical data could be integrated with real time information enabling predictive services which could maintain the value of older data.

Fourth Law of Information: The value of information increases with accuracy – The automatic identification and individualisation provided by IoT devices allows for a fine grained view of the real world (and individual entities, such as people, houses or vehicles). This allows for a far greater level of accuracy in data gathering than historical method of monitoring, therefore providing a greater depth of information for sale to collaborators.

Fifth Law of Information: The value of information increases when combined with other information – For example a SatNav algorithm with just a destination will adequately propose a route for the driver, but if it included information about traffic conditions, weather conditions, driver road preferences and the time of the drivers appointment from his calender it could arguably do a better job. The IoT provides access to multiple data streams from different objects, and the sharing of information between data aggregators and information service providers will increase the value of other information it gets analysed with.

Sixth Law of Information: More information is not necessarily better – More information is better, and increases in value, up to a certain point. When there is more information incoming than can be processed it leads to information overload and a decrease in the usefulness of that information. The IoT provides an inherent level of filtering due to device individualisation, however personalisation and customisation of information through pre-processing can not only reduce the chance of information overload, but potentially add to the value of the incoming information by making it even more usable.

Seventh Law of Information: Information is not depletable – Not only can you not loose the value of information (other than any depreciation over time) but information is self-generating. This occurs through the processing, analysis, summarising or combining with other information, where new information is created from the old. With the vast array of potential device types, and therefore data streams available from the IoT this law should be considered fertile ground for value creation within any business model.

B. Revenue Generation for IoT

With the IoT more (and in particular more detailed) information becomes available, whilst also being more directly associated with individually identifiable devices. This higher level of information granularity provides the ability to trace specific device usage, status and location data, which provides for new types of value proposition scenarios not previously available to the traditional business approaches.

With these new opportunities, such as linking product related data to a consumer (e.g. carbon footprint) or the exact billing of products and services based on actual use (e.g. rental car usage or returnable transport items), comes a need to translate the value of data into value propositions. Bucherer & Uckelman [11] list the following requirements as specific elements that allow the principles described by the Seven Laws of Data to be used for value propositions which take the perspective of fulfilling customer needs:

Providing the right information:

- Linking information to a specific device

… in the right granularity...

- High information granularity, providing new levels of clarity and insight

… and the right condition …

- High information accuracy
- Aggregation of information from numerous sources, such as tags, sensors and embedded systems
- Integration and further analysis of other data allowing for new insights to be derived
- Pre-defined syntax and semantics

… at the right time …

- Timeliness of data
- Access to real time information in addition to historical data for analysis
- Real-time analysis and business intelligence providing high resolution management
- Intelligent real-time decision making capability based on real-time physical events

… anywhere in the network …

- Online access with possible offline usage
- mobile access

… at an appropriate price

- Transparent pricing
- Low premium for billing services – price should be for the information not the infrastructure

With this new perspective on informations potential roles in value propositions comes the opportunity for new revenue streams for businesses. The value of information has typically been determined by the cost of the infrastructure, but now information can (and should for IoT business models) have its value calculation decoupled from the cost of the sensor, tags, networks and hardware used to collect and store it. For IoT businesses the value of information, and the revenues that can be derived from that, should be based upon information flows to customers and the benefits derived by those customers, who may not be the consumers or users of the device.

This introduces another important aspect, that of new customer relationships which can open up new avenues of revenue generation. Compared to the physical exchange of products, where the actors are generally limited to the value chain with revenue generation ending at supply to consumer, information exchange can involve range of involved parties outside those traditional relationships. The relationships with these actors can be identified and defined by information flows which can be unidirectional, bi-directional or multidirectional.

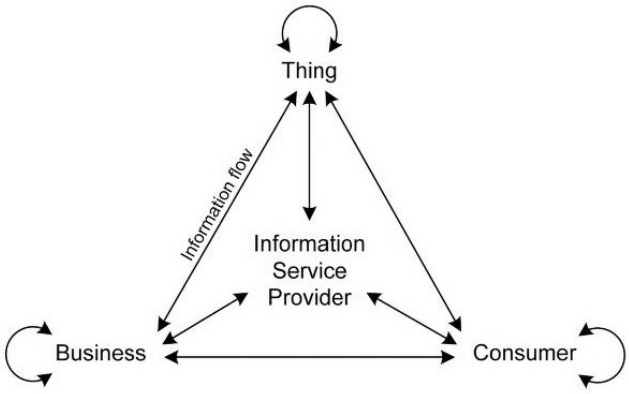

Fig. 3. Information providers and information flows in the IoT [11]

Figure 3 shows the relationships of information producers and consumers linked by information flows. This model can be used to identify potential potential value proposition opportunities and consider new customer relationships not viable under the traditional product based paradigm.

IV. BUSINESS MODELS FOR IOT

Each business activity can be reduced to a number of core elements, which at the simplest level consist of the value proposition, production and distribution, and customers of the company. These aspects describe how the company produces and sells a good or service and, although not always explicitly expressed as such, constitutes a business model which can represent how that business operates. Although 'business model' is a much used term which varies in meaning, a commonly cited definition is found in Timmers (1998) [13]. He describes business models as an architecture of the products, services and information flows which includes the involved actors, their roles, the potential value created for all participants and the sources of revenue.

The concept of business models as a system of components, links between components and their dynamics [14] is pretty universal and there are a wide variety of proposed components and ways to link them within literature. One approach is the Business Model Canvas by Osterwalder and Pigneur (2009) [15] which describe 4 main perspectives, those of value proposition, the customer, financials and infrastructure (figure 4).

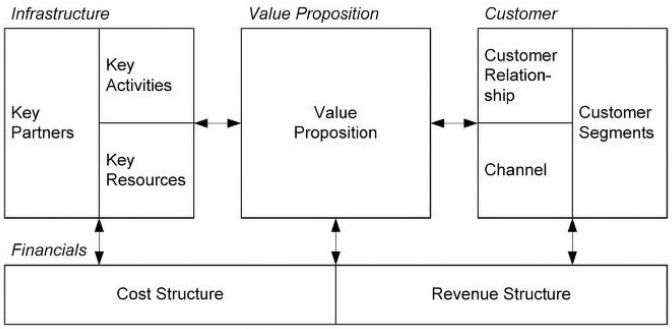

Fig. 4. Business Model Framework – adapted from [15] by [11]

The *value proposition* defines what is actually provided for the customer. It identifies actual customer needs and the aspects of the product or service that directly provide for those needs. It also describes quantitative (price, time of delivery etc.) and qualitative (brand, design etc.) elements which are integral to the value offered. For the IoT information derived from products and the amalgamation of information from numerous sources form the basis for value propositions rather than the product related benefits of traditional business models.

The *customer* component describes the groups of people who are served by the value proposition. It includes the range of *customer segments* (the different groups of potential customers), how those customers are interacted with through *channels* and the *relationships* with those customers. With information as the core value proposition the customers an IoT company might be engaged with could be very different to a traditional product based approach. Although end user consumers may well be one customer segment, new customers who could derive benefit from the information collected, or from the integration of multiple information streams to provide new types of useful information, become viable. This broadening of potential customer types provides significant opportunities for value capture and represents one of the main advantages of applying IoT thinking to planning business strategies.

The *financial perspective* includes costs as well as revenues derived from the value proposition. The *revenue structure* describes the various sources and methods of revenue generation based on the customers identified in the customer component. Here the specific types of revenue stream are identified, such as asset sale, subscription fees, licensing, renting, brokerage fees or advertising. Again, IoT approaches provide new possibilities of how information can be charged for and how those revenue streams are managed, such as a far finer level of control of specific product usage charging (per use / per time unit) than has been possible before.

The *infrastructure* components consist of *key partners*, *key activities* and *key resources*. These elements describe what the business needs to do (activities), use (resources) and who they need to collaborate with (partners) to be able to offer the value proposition. The infrastructure component will dictate how a company needs to structure itself (or re-structure its operation) to align itself with IoT principles. With new types of information based value propositions the way a company needs to operate may be substantially different to traditional approaches, and the range of resources and partners shift, or expand, significantly.

With information now playing a central role as the value proposition, rather than traditional product centered value propositions, we now have a business model which potentially develops in a very different manner to traditional approaches. We might see different types of customer, or relationships with those customers along with new partner relationships and different focusses for company infrastructure aspects. More importantly, this approach can lead to fundamental shifts in how revenue is generated from the value proposition and how the costs of producing / supporting the value proposition are assessed.

Although we have seen significant examples of companies successfully applying new business models to the internet domain (such as Amazon, Google, eBay and Apple iTunes) there are many companies who are not actively transitioning their businesses to encompass these new ways of (and opportunities) of doing business. In 2008 IBM's Global CEO Report [16] indicated that 98% of CEO's saw the need for business model innovation in their companies, with market factors having the biggest effect on their organisations. In 2012 IBM's report [17] technology is (for the first time) considered the largest driver for change in the companies (figure 5), along with showing significant need for a shift in partner collaboration approaches. The report also demonstrates the most successful companies are those most able to manage change.

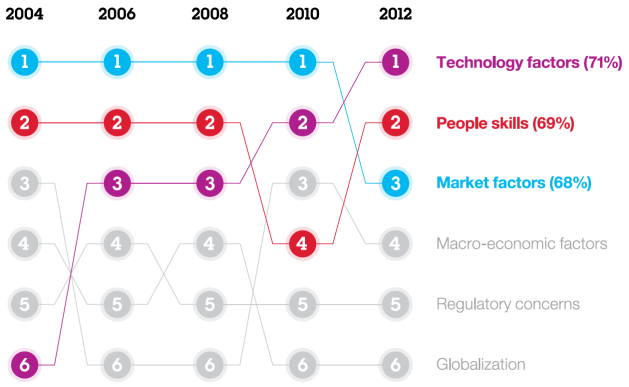

Fig. 5. External forces expected to impact organisations over the next 3-5 years [17]

This requirement for change in companies approaches to an increasingly competitive and dynamic market place, combined with the rise in importance of technology, big data and new collaborations, demonstrates the need for structured approaches to guide and manage such changes within companies. Business models with an alignment to IoT principles have the potential to offer companies a holistic approach to assessing business opportunities and how to engage with them in the new business environments presented with the growth of IoT.

V. NEST

Nest Labs [12] is a company that designs and produces thermostats and smoke-detectors for home automation. Their products are programmable, sensor-driven, WI-FI enabled and self-learning devices which can be remotely controlled. Their main product, and the focus of this section of the paper, is the Nest Learning Thermostat.

A. The Nest Thermostat

The Nest Thermostat (Figure 6) is used to control a properties heating and cooling systems. It is a round device mounted on the wall sporting a round LCD display and outer control ring which the user turns to adjust the temperature. The device includes sensors for temperature, humidity, proximity, far-field activity, near-field activity and ambient light, along with Wi-Fi support for 802.11b/g/n and 802.15.4 Wi-Fi (both at 2.4 Ghz) [13].

Fig. 6. The Nest Learning Thermostat

Although the Nest thermostat is programmable for set heating ranges and schedules the device is able to learn the householder's preferences and patterns to generate its own schedule. Over the first week the user just adjusts the temperature manually to the desired setting (turning it down when leaving the house or going to bed). The Linux based software employs a machine learning algorithm to construct a schedule of temperatures and times which it will then automatically adjust the heating too.

In addition to the automated learning functionality the device uses its motion sensors to detect when the house is empty. When no motion has been detected for two hours the thermostat engages its 'Auto-Away' mode and automatically reduces the temperature to avoid heating an empty home.

B. Product Focused Business Model

With estimated sales of 200,000 units a month in 2014 1[20] and with estimated revenues of $300M the Nest thermostat is obviously a product that can take centre stage in a solid business model. Funk (2014) [21] examines the Nest Thermometer business model from a product centred perspective.

From this perspective the Nest Thermometer offers ends users the *value propositions* of reduced energy consumption (with associated costs savings), ease of use (self-programming and automated), remote control (via mobile apps), aesthetics and lifestyle symbols. These value propositions are intrinsically 'per product' based and entirely focused on home owner consumers of the product. As such the *customer perspective* is mainly focused around homeowners, with any collaborations, such as with new home construction companies, utility service providers and government programs, aimed at just getting more individual units sold and installed.

The value generation is limited to the *revenue stream* of unit sales and control app sales associated with each device. Estimates of revenue using this business model range from $375M [21] to $412M [20] by 2015. This approach is an

excellent example of the 'one-and-done' paradigm of selling product, where for Nest Labs to continue making money they have to keep selling new thermostats. This leads to the business being reliant on developing new models or products that people will keep on buying (and how often will homeowners want to replace a heating thermostat?).

In January 2014 Nest Labs was acquired by Google for $3.2 billion [22]. To recoup this sort of purchase price a significant number of products would need to be sold if just considering a product centric business model. Now withstanding the fact that Nest have other products in their portfolio (smoke alarms and security cameras), and the discussion about Google buying their way into a new market place whilst paying a premium to maintain the old management structure, there are commentators who consider Google may have overpaid for Nest [20].

An alternative perspective is that there is a whole layer of value contained in the Nest company which does not get explained (or accessed) by this product based mindset. It is possible that the real value of Nest Labs lies in an IoT oriented business model which introduced whole new value propositions, customer relationships and revenue streams.

C. IoT Focused Business Model

Nest Labs have developed a business model that uses the data collected by their thermostats in homes at its centre rather than the sale of the product itself. Nest sells information on energy usage patterns to electricity utility companies which allows them to plan their energy production more effectively [23].

In electricity supply any over production cannot be stored and so goes to waste, however too little production means there's not enough to go round when needed and leads to brown-outs (the reduction or restriction of supply in a particular area). Utility companies have to predict future usage and plan their production or purchasing of electricity accordingly. Their objective is to have just a slight overproduction to actual usage to minimise any waste.

In the past energy producers have had little more than historical data of usage to predict future demand and plan their production on. With intelligent products like the Nest Thermometer it is now possible to get far more immediate and more detailed information on actual usage by its customers. Indeed, with the Nest's profiling capability, it is even possible to get detailed information on the expected usage as every thermometer has the information of when it expects to turn up the heat (or air conditioning) depending on its knowledge of that families activity patterns (i.e. when they get home from work).

These usage patterns (both real-time usage and future predictions) represent valuable information to utility companies who can use it to make more accurate prediction regarding production, reduce over production and consequently save significant amounts of money. Nest have built a relationship with a number of energy providers where they are being paid a fee for the supply of usage metrics, in the order of $40 per installation per year.

When we examine the information provided by Nest to the utility companies we can see how it aligns to revenue generation (section II B) based on the 7 Laws of Information discussed above. Providing the *right information* consists of accurate usage statistics linked to real homes identified uniquely by their Nest Thermometers ID. This information has a high level of *granularity*, especially compared to the regional usage metrics electricity companies had to rely on previously. The *right condition* of the information is represented by high accuracy, combined metrics from multiple thermometers in one building and predictive usage data. The real-time nature of usage data, combined with the access to historical and future predictions gives utilities access to '*right time*' information, and accessible from *anywhere* through web connectivity. Finally the *appropriate pricing* is provided by transparent prices per installation where the cost of information is decoupled from the cost of the infrastructure.

With a clearer picture of the nature of the value the information the company possesses it now can take a core role as a value proposition in an IoT focused business model. Figure 7 uses the Business Model Framework presented above to construct a business model for the Nest Thermometer using an IoT centred approach.

In the business model for the Nest Thermometer we can see the real-time energy usage information and predictive analytics providing the information based *value propositions*. The *customer component* now consists of utility companies being the consumer of the service accessing it via web connections through automated systems. The *revenue structure* provides for per installation subscription fees rather than a per product sales revenue, with costs being associated with data collection and processing rather than costs of unit production. Finally the *infrastructure component*, with related key activities, resources and partners, acts as a guide to how the company needs to structure its activities and partner relationships in light of an information based value proposition.

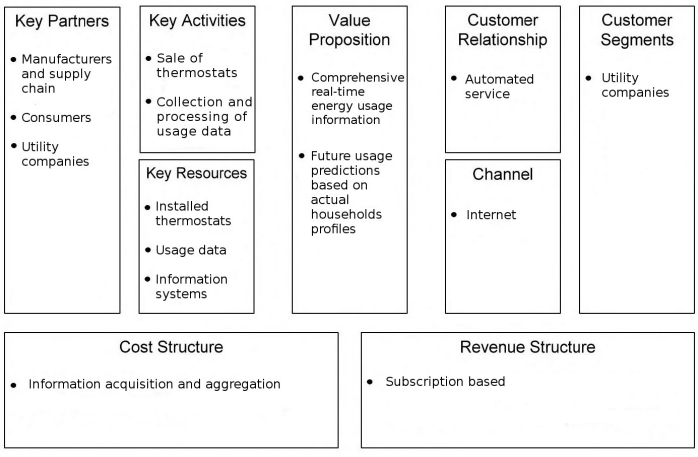

Fig. 7. Business Model for the Nest Thermometer

In early 2014 Nest had 12 major utility partners [24] using this subscription model with an expectation that number would grow in the future, with Nest's founder Tony Fadell stating he

expected this side of the business will in time earn more for the company than the sale of devices [23]. There is also the potential for Nest to investigate customer segments other than utility companies, such as renewable energy companies like solar panel companies for the monitoring and control of distributed energy technologies rather than just grid-based energy supply.

Nest have also recently announced trials where they control users house temperatures through their Nest Thermostats at the request of utility companies to manage peak usage [25]. In parts of America, on hot days, there can be a problem with numerous air conditioners being turned on at the same time (when people return from work for instance) and the demand outstrips the energy companies production capabilities. This forces the company to buy extra electricity on the energy market (along with many other supply companies who are in the same position) which pushes the price of electricity up a huge amount, or enforce restrictions in supply (brown-outs).

The principle behind the trial being conducted by Nest and a number of utility companies is that a small reduction in usage by a large number of people at those critical times of day can make a significant difference to peak demand. To achieve this home owners are offered a scheme, called Rush Hour Rewards, that they can subscribe too which gives them discount rewards on their electricity bills if they allow Nest to take control of their thermostats during peak load.

Nest uses an algorithm combining local weather conditions, the actual temperature of the house, the customers usual schedules and the preferred temperatures of that customer. This allows them to micro-manage individual houses temperatures and gradually reduce them by just a degree or two so the occupant does not notice the shift. Another approach is to predict a demand spike and automatically pre-cool the house (before the owner gets home) using cheaper electricity before the period of high demand occurs.

The Rush Hour Rewards trial demonstrates how new value propositions can be leveraged from the same data and how value can be offered to numerous customers at the same time (here benefiting homeowners with the opportunity of discounts and utility companies with the management of peak demand). It also demonstrates the power IoT enabled devices offer where multiple information streams are processed and real-time control decisions are made to manage individual home's temperatures for the benefit of another customer.

VI. CONCLUSION

Aligning business operations to the new marketplace expected with development of IoT devices and infrastructures patently has importance for the future of companies. With a structured approach of considering information as value propositions it is possible to employ business modelling as a tool to design (or re-design) a companies structure to operate within, and exploit, the new types of opportunities the IoT offers.

The Nest example clearly demonstrates the principles presented in this paper of attributing value to information produced by IoT devices and converting those value concepts to revenue generation principles which can be used as value propositions in a business model.

The Nest Thermostats IoT business model described above shows how new customer segments can be defined when information value propositions are used, along with completely different revenue models from that of the 'one-and-done' physical product revenue stream. It also demonstrates how the landscape of business partners can shift and how the operating costs and infrastructure can change in light of a new business approach.

As such, the Nest example acts as a validation of the approach presented by this paper of using information and its value aspects as a core foundation for creating business models which are effective at aligning companies to IoT.

The author considers that, not only is there potentially great value for companies to start considering their business model in light of IoT, but also that a framework focussing on information (such as the one presented here) is the key to the required shift in perspective.

VII. SUMMARY

With the expected drastic increases in IoT enabled devices there is a need for companies to prepare for business in a market place where information takes on a far larger and more complex role than ever before. It is arguable that this will require a shift in mindset from a traditional product based approach to one that considers information, and the value of that information, as a central component of business structure and operations.

Business Models offer one structured way for companies to realign their offerings and operations with an IoT way of thinking which places information at its core. This requires information to be considered and understood as an asset with intrinsic value which can then be used as value propositions for an IoT oriented business model.

This paper has presented a method of ascribing value to information and a Business Model Framework that can be used to formulate suitable business models. An example of the Nest Thermometer is used to demonstrate the principles of an information centred business model and how IoT focused approaches can provide new business opportunities.

It is hoped that this examination of business modelling for IoT can be used as a useful introduction for companies who might not be aware of the implications IoT might have on them, and to act as a guide to how companies might go about investigating the integration of IoT principles into their business planning for the future.

REFERENCES

[1] P. Fan and G. Zhou, “Analysis of the business model innovation of the technology of internet of things in postal logistics,” *2011 IEEE 18th Int. Conf. Ind. Eng. Eng. Manag.*, pp. 532–536, Sep. 2011.

[2] Harbor Research, “Smart Services and Internet of Things Business Model Innovation,” 2014. [Online]. Available:

http://harborresearch.com/smart-services-and-internet-of-things-business-model-innovation/. [Accessed: 22-Oct-2014].

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[4] S. Leminen, M. Westerlund, M. Rajahonka, and R. Siuruainen, "Towards IOT ecosystems and business models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7469 LNCS, pp. 15–26.

[5] J. Rivera and R. van der Meulen, "Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020," 2013. [Online]. Available: http://www.gartner.com/newsroom/id/2636073. [Accessed: 22-Oct-2014].

[6] O. Kharif, "Cisco CEO Pegs Internet of Things as $19 Trillion Market," 2014. [Online]. Available: http://www.bloomberg.com/news/2014-01-08/cisco-ceo-pegs-internet-of-things-as-19-trillion-market.html. [Accessed: 22-Oct-2014].

[7] H. Lee and J. Kwon, "Combining context-awareness with wearable computing for emotion-based contents service," *Int. J. Adv. Sci. …*, vol. 22, pp. 13–24, 2010.

[8] V. Lipman, "71% Of 16-To-24-Year-Olds Want 'Wearable Tech.' Why Don't I Even Want To Wear A Watch?," 2014. [Online]. Available: http://www.forbes.com/sites/victorlipman/2014/09/22/71-of-16-24s-want-wearable-tech-why-dont-i-even-want-to-wear-a-watch/. [Accessed: 22-Oct-2014].

[9] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[10] G. Hui, "How the Internet of Things Changes Business Models," 2014. [Online]. Available: http://blogs.hbr.org/2014/07/how-the-internet-of-things-changes-business-models/. [Accessed: 22-Oct-2014].

[11] E. Bucherer and D. Uckelmann, "Business Models for the Internet of Things," in *Architecting the Internet of Things SE - 10*, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. Springer Berlin Heidelberg, 2011, pp. 253–277.

[12] D. Moody and P. Walsh, "Measuring the value of information: an asset valuation approach," in *Guidelines for Implementing Data Resource Management. 4th edn.*, 2002.

[13] P. Timmers, "Business models for electronic markets," *Electron. Mark.*, no. 8, pp. 3–8, 1998.

[14] A. Afuah and C. Tucci, *Internet business models and strategies: Text and cases*, 2nd ed. New York: McGraw-Hill, 2002.

[15] A. Osterwalder and Y. Pigneur, *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. New Jersey: Wiley, 2010.

[16] IBM, "IBM 2008 Global CEO Study," 2008. [Online]. Available: https://www-935.ibm.com/services/uk/gbs/pdf/ibm_ceo_study_2008.pdf .

[17] IBM, "IBM 2012 Global CEO Study," 2012. [Online]. Available: www-935.ibm.com/services/multimedia/anz_ceo_study_2012.pdf . [Accessed: 02-Nov-2014].

[18] "Nest Labs." [Online]. Available: https://nest.com/uk/. [Accessed: 25-Oct-2014].

[19] "2nd generation Nest Learning Thermostat technical specifications," 2014. [Online]. Available: https://support.nest.com/article/2nd-generation-Nest-Learning-Thermostat-technical-specifications. [Accessed: 25-Oct-2014].

[20] Nextmarket.co, "Breaking Down The Valuation For Nest's $3.2 Billion Purchase Price," 2014. [Online]. Available: http://blog.nextmarket.co/post/73320622998/breaking-down-the-valuation-for-nests-3-2-billion. [Accessed: 01-Nov-2014].

[21] J. Funk, "Biz Model for Nest's Smart Thermostat," 2014. [Online]. Available: http://www.slideshare.net/funk97/biz-model-for-nests-smart-thermostat. [Accessed: 01-Nov-2014].

[22] Google Investor Relations, "Google to Acquire Nest," 2014. [Online]. Available: https://investor.google.com/releases/2014/0113.html. [Accessed: 02-Nov-2014].

[23] M. Mombrea, "Google's real plan behind the purchase of the Nest thermostat," 2014. [Online]. Available: http://www.itworld.com/article/2833423/big-data/google-s-real-plan-behind-the-purchase-of-the-nest-thermostat.html. [Accessed: 02-Nov-2014].

[24] C. Cameron, "The future of Google's Nest," 2014. [Online]. Available: http://www.utilitydive.com/news/the-future-of-googles-nest/257068/. [Accessed: 02-Nov-2014].

[25] S. Levy, "Nest's Plan to Stop Brownouts Before They Start," 2014. [Online]. Available: http://www.wired.com/2013/04/nest-energy-services/. [Accessed: 02-Nov-2014].

Using metrics to improve design in free-to-play games

Henna-Riikka Ruonala
University of Helsinki
Helsinki, Finland
Email: henna-riikka.ruonala@helsinki.fi

***Abstract*—A literature review on metrics use in free-to-play games was undertaken, searching for common metrics in use for the development, monitoring, and improvement of free-to-play games. The most significant metrics found were related to measuring player engagement and player churn. Other gameplay related metrics tend to be more game specific, related to the game mechanics and genre.**

I. Introduction

The freemium or free-to-play model has gained ground in recent years in the games industry. Players are able to start playing the game without having to pay anything. Often they can keep playing for free, but there are certain items, bonuses, or in-game currency that one can buy from within the game. This might lead to faster progress through the game, an advantage over other players in a multiplayer setting, or just an aesthetic change. The majority of the players never pay anything, but if the number of players is large enough, there are also enough paying customers. The aim is to get enough players to try the game out, and then to keep a good percentage of them coming back as well. As Drachen et al. [1] point out, free-to-play games need metrics data to keep the players playing, and consequently also spending money. This leads to regular monitoring and updating of the game. The updates need to be spaced so that people keep being engaged with the game, and the update cycles can also drive players to come back to the game more often, so that they do not miss timed content [2].

The themes, genres, and gameplay mechanics of free-to-play games are quite varied, as free-to-play is only a description of the economic model of the game. Thus a free-to-play game can be a casual, social Facebook or mobile game, an online roleplaying game, a real-time strategy game, a first-person shooter, a digital trading card game, etc. The common thing among all these different types of games is only that they are offered with free entry, more as services rather than products. Getting the game released for the public is only the beginning of the development process. This makes gathering data on the success of the game design even more important than in games sold as products. Many of these points and the metrics described in this paper are also applicable to subscription based persistent games.

In order to provide a steady stream of updates, the game developers need to know which direction the game design should take. Quantitative and qualitative feedback can help in identifying game design flaws or in experimenting with new features. Qualitative feedback can be gathered through player surveys, and quantitative gameplay data can be gathered as players play the game. Both of these complement each other and help in interpreting the data. This paper will take a look at quantitative or telemetry data, and how to make use of it in a free-to-play context. It will however not go into the details of how the data is gathered or analysed.

There are various types of metrics that can be used to analyse gameplay and other factors of a game. The choice of gameplay metrics naturally depends on the type of game in question. There are however some metrics that can be gathered regardless of the game's genre or mechanics. The aim of this paper is to gather together some existing and known practices in applying metrics analysis in free-to-play games, and to give some examples of useful gameplay metrics. First some related work in free-to-play games and game metrics research is looked at briefly in section II, followed by a description of the research method of this paper in section III. Then the results are presented in section IV and finally some further issues and implications are discussed in section V.

II. Related Work

So far there has been very little work looking into metrics use in free-to-play games specifically. There have been some studies on other aspects of free-to-play games however, especially regarding the social aspects of some of them. Some more data is available on gameplay metrics, but these are often from the perspective of playtesting first-person shooters or role-playing games, and detailing the technical aspects or visualisation of the telemetry data. Some general points pertaining to gameplay metrics can nevertheless be useful in a free-to-play scenario as well.

A closely related field of study is game user research, which is often interested in answering the same type of questions regarding e.g. player engagement, but deploys a variety of methodologies. Game metrics analysis is one of them.

A. Free-to-Play Games

Sotamaa et al. [2] have taken a look at Zynga's Frontierville, a social free-to-play Facebook game that is free to play with fast update cycles. They highlight the importance of rhythms for social games. The rhythms are created by game mechanics (timers limiting how often the player clicks on things), session length, weekly updates, and more sparse seasonal theme updates.

A work-in-progress paper by Paavilainen et al. [6] recognised user interface layout, navigation and the help system as key areas with the most playability issues in free-to-play social games. If a game has playability issues, it will also affect the rate at which players will leave the game, switching perhaps to another free-to-play game. Players who are not satisfied with the game will not spend money on it either, even if they keep playing.

B. Game Metrics

As Drachen and Canossa [7] point out, gameplay metrics analysis is focused on looking at what the player does in the game. Additionally, hypotheses can be made about why the player is behaving in a certain manner. To back up the hypotheses, other methods should be used with the metrics. Gómez-Maureira et al. [8] did a comparison of methodologies, combining gameplay metrics, biometrics, and interviews to inform improvements on level design in a 2D platformer. They found that data from all three methodologies were needed for best results. For a level designer, each of the combinations of two methods offered suggestions for improving level design, but there were differences in which things were suggested. The differences were however not as large as suspected initially. The study nevertheless indicates that it is important to back the gameplay metrics data with some other source, especially in determining why players are doing what they are doing. The advantage of gameplay data is that it is readily available, as long as there is room to store the data.

Lanzi et al. [9] gathered gameplay data including user gestures and collisions of game objects from two playtests. The metrics showed a problem with the pace of the game, so the game design was updated according to the results.

C. Game Metrics in Free-to-Play Games

Gagné et al. [3] have analysed telemetry data from a free-to-play real-time strategy game, Pixel Legions. They worked with the developer to create a system of visualisation for the gameplay data to help answer the game design questions and hypotheses the developer had.

Hadiji et al. [4] found that predicting player churn can be done based on the patterns of player logins and other session data, which are available for any type of game. Another study [5] also attempted to predict high-value player churn, as well as testing incentives for players to return to the game.

III. Research Method

The research method applied is a literature review summarising metrics use in the context of freemium games. Because of the sparsity of articles on the topic, various search terms such as game analytics, game data mining, free-to-play, etc. were used to uncover possible articles of interest in different article databases. Further articles were found also by going through the references listed in the found articles. Some additional non-academic articles and blog posts were used to gather a more complete view of game analytics as it is in use in the industry.

IV. Results

Perhaps the most important and used metrics in free-to-play games are related to player engagement and player churn. The two are closely related, as to avoid churn, one needs to engage the players. Game developers are interested in finding out how engaging the game is during a certain period of time (for example after an update, or an advertisement campaign), and in predicting players that are about to churn, i.e. leave the game. These churning players could then be contacted and incentivised to stay.

Other metrics related to gameplay can be used to guide design decisions for updates. These metrics are more varied across game genres.

A. Player Engagement

The metrics for free-to-play games are by necessity strongly tied to monetisation, but to be able to monetise, the game needs to engage players. The critical point in the game for player engagement is the beginning. Players need to be guided through the first hurdles of learning the game controls and gameplay mechanics, so that they will keep playing and coming back to play. Gathering metrics on the tutorial or the first levels of a game is especially important for its success. In the end player engagement is the driving force behind the use of telemetry data and the metrics derived from it for improving game design. Metrics for measuring player engagement include:

- DAU/MAU
- sessions/DAU

The simplest and most important metric for measuring player engagement is the ratio of daily average users (DAU) per monthly average users (MAU), i.e. the DAU/MAU ratio. Game design changes can be reflected in this ratio, and if the ratio is too low, efforts should be directed into improving the game design [10]. Some quote 20% as a good percentage [11].

There are some things to take into consideration when using the DAU/MAU. It is a constantly changing ratio, and affected by the amount of new users coming in because of an advertisement campaign, for example [11]. Another thing to remember is that if the percentage of players who keep coming back to play is low, it in itself does not tell anything about why it is low.

A related metric also measuring player engagement is sessions/DAU, meaning how many times players log in to the game per day [12]. The exact number varies according to game genre and how long the average session in the game is. The sessions/DAU ratio is nevertheless more stable than DAU/MAU [11]. Thus if sessions/DAU starts declining, something has gone wrong.

The length and pacing of sessions can also be used to guide design decisions. Developers may want to favour a certain style of gameplay, tracking what seems to be the most popular session length or number of sessions per day [13]. Junebud for example tracks session length and frequency by recording login and logout times for each player [14]. Session length and other data was also tracked in Pixel Legions [3], to find out aspects of player behaviour such as:

- When players stop playing
- How often they lose
- What players care about (the kind of levels they prefer etc.)
- How they handle difficulty
- Which levels players like

This (session data) combined with the tracking of player progression will inform developers of update needs, so that players do not run out of content. Even game mechanics can be introduced that will take care that players do not progress too fast through the entire content [2]. This includes the use of timers or energy bars, which were employed in e.g. Frontierville to great extent, creating a steady rhythm for the game. At the same time these can of course be used to entice players to spend money.

B. Player Retention

The retention rate describes the percentage of players that stay and play the game again over a certain period of time. It is calculated per installation of the game, from the installation date. Day 1 retention gathers up the players who came back the next day after installing the game, day 7 retention has all the players who came back (any day) within a week of installation. A good retention rate for day 1 is thought to be 35-40

C. Churn Prediction

As the retention rate measures how well players are staying with the game, the opposite is true of the churn rate, which tells how many users are lost during a period of time. Game developers are interested in trying to predict this rate, and to identify the potential churners, the players who leave the game and do not return. Hadiji et al. [4] have developed a churn prediction model independent of game design. By using decision trees, and using a more relaxed definition of churn, they were able to predict which players were about to leave the game. Instead of looking at players churning by a fixed point in time, the study made use of a window, in which the players that were about to churn were already considered churned. This allows for time for the developers to motivate the players to return. The study found that the average time between sessions was an important predicting feature. This helps towards an understanding of why players churn.

A contributed article [15] on Gamasutra, a website dedicated to game development, details data mining efforts for churn prediction in the MMO game Aion. Even though a neural network approach was working well, they preferred decision trees. They were able to get the prediction rate of decision trees up and false positives down by refining the model by choosing activity and game-specific metrics which were listed as follows:

- Playtime at current level, previous level, and total during lifetime
- Mobs killed per minute (current/previous/lifetime)
- Quests completed per minute (same)
- Average playtime per play day
- Days of play
- Absenteeism rate (number of days skipped during the seven day free trial)

Despite the hypotheses in the beginning and managing to make the prediction more accurate, no specific features were found to be the reason behind why players were churning in general. The data for individual players was nevertheless useful in finding ways to incentivise the player to return. An email was sent before the predicted churn, giving useful advice on the particular topics that player might have been interested in or struggling with.

In both churn predictions, the high rate of people leaving in the beginning of the game has a negative impact on the ability to predict churn. In the case of Aion, this was contained into only the first level of the game by predicting churn rates for each level separately [15]. Thus even though the beginning of the game is significant in keeping players interested, it is difficult to make predictions for it. Interviews and surveys can perhaps shed more light on why players quit so soon [6]. Gameplay metrics analysing if the players are playing the game as intended, taking the tutorial, etc. can also help in a more limited manner.

Another study [5] in churn prediction focused on so called high-value players specifically. They also found that a neural network works best for prediction. Included in the study was an A/B test for determining best ways of getting players back to the game. It was perceived that incentives like free in-game currency did not help to re-engage players. Email and Facebook notifications were however better received before the churn event rather than after, making the churn prediction useful. Another question is what the email or notification should contain. In the case of Aion, customised advice was given to the players [15], whereas in the case of high-value players leaving, it might be better to direct them to another game from the company [5].

The study on high-value players [5] also found that there are differences between game genres and types and the ability to predict player churn. Prediction is easier for a game that requires more interaction and different types of interaction, and thus more regularly spaced logins.

D. Gameplay Metrics

Whereas the metrics measuring player engagement or non-engagement are not very helpful in answering why things are as they are, gameplay metrics can yield more information about what the players are actually doing. If interpreted correctly, they can even begin to answer why player engagement is low, for example.

Playtime, player's progress through levels, use of items, etc. are generally good to track in any game [16]. Playtime is related to user engagement and the previous metrics described above. Progress through levels lets the developer know if the game is getting difficult for the players at a steady pace, or if there are unexpected tough spots. Players could be employing a wrong strategy or choosing the wrong route for example, and could perhaps be better guided towards the correct choice. The use of items has special significance in competitive games, where the balance between different items is important. If a

certain weapon is used much more than others, it might mean that the weapon is overpowered compared to the rest of the available weaponry.

The balancing of the game is important in free-to-play games as well, as players get frustrated if they feel the game is not fair. It is especially important for user satisfaction to avoid a complete pay-to-win scenario, where one can buy more powerful items or bonuses with real money and thus make better progress in the game than others who do not spend as much money [17]. There might be cultural differences in to which extent players are willing to accept paying to win, as in China the phenomenon is apparently much more widespread [17].

A good pace of action is important for player engagement. In mobile games, the gameplay data worth tracking in an action based game is related to the player's gestures on the touch screen, and the collisions between objects. This was studied by Lanzi et al. [9] by conducting two playtests during public events. A flaw in the pace of the game was discovered and corrected. The player's taps on the screen were too evenly distributed, making the game boring to play for a longer time. The solution was to add boss fights to break the monotonicity. The boss fights lead to an intense period of tapping on the screen, leading to better engagement.

Game metrics analysis allows monitoring of things such as over- or underused game world areas, game features used as intended, and barriers to player progression [16]. This was done in a case study Hullet et al. [18] where they found several unpopular features in a console game (Project Gotham Racing 4) by analysing data e.g. on which gameplay modes players were choosing. While not a free-to-play game, these findings highlight the importance of pruning the content, especially as there is a need for regular updates in a free-to-play scenario. Too much extra content will slow down the update cycle and bring in new possibilities for bugs.

V. Discussion and Implications

The type of analytics that are used to analyse free-to-play games are by necessity often focused on the monetisation of the game, measuring the percentage of paying players. Even though monetisation is important, Zynga for example has realised through experience that it should not drive the changes in the game [19]. Some of the metrics can nevertheless give information on how well changes were received. Such metrics are related to player engagement, such as the DAU/MAU ratio, and information on play session length and frequency. These were the metrics perceived to be relevant to this literature review. Other metrics that are more related to the economics or monetisation of the game (such as player lifetime value) were left out as they were generally perceived to offer little information on the gameplay experience or player satisfaction.

The metrics with clearest information on the success of the game design are the ones directly related to gameplay, such as player paths or gestures, or the frequency of taps (as in the mobile game example by Lanzi et al. [9]). Metrics on player engagement are more suited for testing out hypotheses, as one only gets a rough estimate of what people thought of the update by analysing the player numbers. Churn prediction can offer a bit more information on the point at which players are leaving the game, offering time or level based clues on which parts to improve.

Whereas the data related to the amount of users and sessions can simply be expressed in numbers, gameplay data is often more useful and understandable when it is visualised. This is especially true of spatial and geographical data, like player paths, or the player's interaction with the user interface of the game. It is important to remember that the data mined and the metrics used must be made accessible for the developers and game designers. This is another important topic that was left out from this review. An example of a data visualisation built for the whole team is Data Cracker for the game Dead Space 2 [20].

It seems most of the research on free-to-play games is related to the social aspects of many of those games. Some articles also were found focusing on churn prediction in free-to-play or persistent games. Research on gameplay metrics is mostly focused on playtesting or game user research, while game user research offers a more complete look into game metrics, adding qualitative data like surveys to help interpret the quantitative data.

The focus of this review was on free-to-play games, but persistent games in general share a many of the same attributes regarding data gathering. Subscription based games are also interested in churn prediction, for example. Free-to-play games differ from these in their need for monetisation, and thus they need to find a balance in the game design by making the game fun, but not too fun. Free-to-play games are also often characterised by a need for frequent updates, and regularly adding new content. It seems there is very little research focused on this kind of agile or lean game development, especially regarding metrics use, but it would definitely be of interest as more and more developers are starting to use a games-as-service model with fast delivery.

As Sotamaa et al. [21] point out, game metrics and the vast amounts of data available to the developers through them, are changing the relationship between the developer and the player: the feedback from players is automated and there is a deeper understanding of what the players are doing. Similarly the game company Junebud is using the instant feedback to test design hypotheses with each update [14].

There are nevertheless constraints regarding the gathering of telemetry data from persistent games. Some limitations are listed by Zoeller [22]. These include:

- game performance issue
- scalability of the storage needed for data collection
- legal and privacy issues

This has an effect on the type and amount of data that can be gathered. There are limitations to what can be deduced from the data as well. To overcome the problems of interpretation of the data, game metrics can be supplemented by other methods, such as user surveys and biometric data. These methods are nevertheless more costly and need player involvement. As it is, metrics cannot predict which steps to take in the design in the game per se, but they can help in testing out hypotheses.

Despite these concerns, game developers will certainly want to gather some data on how their design is working and

being received. As the game industry is constantly changing, it will remain to be seen what role free-to-play games and games as a service will take. It is nevertheless clear that it is a model that is enticing for both game developers and players as well.

REFERENCES

[1] A. Drachen, C. Thurau, J. Togelius, G. N. Yannakakis, and C. Bauckhage, "Game data mining," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 12, pp. 205–253.

[2] H. Tyni, O. Sotamaa, and S. Toivonen, "Howdy pardner!: On free-to-play, sociability and rhythm design in frontierville," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, ser. MindTrek '11. New York, NY, USA: ACM, 2011, pp. 22–29. [Online]. Available: http://doi.acm.org/10.1145/2181037.2181042

[3] A. R. Gagné, M. Seif El-Nasr, and C. D. Shaw, "Analysis of telemetry data from a real-time strategy game: A case study," *Comput. Entertain.*, vol. 10, no. 3, pp. 2:1–2:25, Dec. 2012. [Online]. Available: http://doi.acm.org/10.1145/2381876.2381878

[4] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage, "Predicting player churn in the wild," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, Aug 2014, pp. 1–8.

[5] J. Runge, P. Gao, F. Garcin, and B. Faltings, "Churn prediction for high-value players in casual social games," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, Aug 2014, pp. 1–8.

[6] J. Paavilainen, H. J. Korhonen, and K. Alha, "Common playability problems in social network games," in *Proceedings of the Extended Abstracts of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI EA '14. New York, NY, USA: ACM, 2014, pp. 1519–1524. [Online]. Available: http://doi.acm.org/10.1145/2559206.2581336

[7] A. Drachen and A. Canossa, "Towards gameplay analysis via gameplay metrics," in *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, ser. MindTrek '09. New York, NY, USA: ACM, 2009, pp. 202–209. [Online]. Available: http://doi.acm.org/10.1145/1621841.1621878

[8] M. A. Gmez-Maureira, M. Westerlaken, D. P. Janssen, S. Gualeni, and L. Calvi, "Improving level design through game user research: A comparison of methodologies," *Entertainment Computing*, no. 0, pp. –, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1875952114000299

[9] P. Lanzi, D. Loiacono, E. Parini, F. Sannicolo, D. Jones, and C. Scamporlino, "Tuning mobile game design using data mining," in *Games Innovation Conference (IGIC), 2013 IEEE International*, Sept 2013, pp. 122–129.

[10] T. Fields, "Game industry metrics terminology and analytics case study," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 4, pp. 53–71.

[11] T. McCalmont. (2013) Muddled mobile metrics. Retrieved on November 7th 2014. [Online]. Available: https://nativex.com/blog/muddled-mobile-metrics/

[12] ——. (2013) How do I know I have a healthy game? Retrieved on November 5th 2014. [Online]. Available: http://www.gamasutra.com/blogs/TrevorMcCalmont/20130228/187460/How_Do_I_Know_I_Have_a_Healthy_Game.php

[13] C. Thurau. (2013) Game metrics: Their true nature and what you shouldnt live without. Retrieved on November 8th 2014. [Online]. Available: http://blogs.unity3d.com/2013/12/05/game-metrics-their-true-nature-and-what-you-shouldnt-live-without/

[14] A. Canossa, "Interview with Ola Holmdahl and Ivan Garde from Junebud," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 30, pp. 689–693.

[15] D. Nozhnin. (2012) Predicting churn: Data-mining your game. Retrieved on November 5th 2014. [Online]. Available: http://www.gamasutra.com/view/feature/170472/predicting_churn_datamining_your_.php

[16] A. Drachen, A. Canossa, and J. R. Møller Sørensen, "Game data mining," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 14, pp. 285–319.

[17] C. Onyett. (2012) Separating free-to-play and pay-to-win. Retrieved on November 7th 2014. [Online]. Available: http://www.ign.com/articles/2012/08/13/separating-free-to-play-and-pay-to-win

[18] K. Hullett, N. Nagappan, E. Schuh, and J. Hopson, "Empirical analysis of user data in game software development," in *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*, Sept 2012, pp. 89–98.

[19] M. Seif El-Nasr, and A. Canossa, "Interview with Jim Baer and Daniel McCaffrey from Zynga," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 5, pp. 73–82.

[20] B. Medler, M. John, and J. Lane, "Data cracker: Developing a visual game analytic tool for analyzing online gameplay," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2365–2374. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979288

[21] O. Sotamaa, H. Tyni, S. Toivonen, T. Malinen, and E. Rautio, "Emerging game cultures," in *New Paradigms for Digital Games: The Finnish Perspective, Future Play Project, Final Report*, ser. TRIM Research Reports. Tampere, Finland: University of Tampere, 2011, no. 3, ch. 4, pp. 31–38.

[22] G. Zoeller, "Game development telemetry in production," in *Game Analytics*, M. Seif El-Nasr, A. Drachen, and A. Canossa, Ed. London, UK: Springer-Verlag, 2013, ch. 7, pp. 111–135.

On Goal-Oriented Requirements Engineering

Nikolay Vasilev
Department of Computer Science
University of Helsinki
Helsinki, Finland
Email: nikolay.vasilev@cs.helsinki.fi

***Abstract*—Goals in software development address the questions why a software product is being built and what the product's ultimate objective is. They can be very useful as a starting point in the extraction and elaboration of more concrete requirements to facilitate the software development process.**
In this investigation we attempt to define the concept of a goal, its significance in requirements analysis in particular and software engineering in general. We further review approaches proposed in literature for modelling and harnessing goals and compare how they treat goals as objects of interest. Finally we draw conclusions about the methods' pros and cons and ponder how we can exploit their strengths in one unified method.

Keywords*—Goal analysis, Goal modelling, Goal-oriented requirements engineering.

I. Introduction

The presence of goal orientation in modern Requirements Engineering (RE) is nowadays tangible [1]. Indeed, attempting to answer the question why a software system is constructed, goals naturally lead to more concrete software requirements [2]. Thus they play an integral part in eliciting, refining, modifying as well as documenting requirements [3]. As such, goals also play a crucial part in use cases in object-oriented approaches.

Over the past twenty years goals have received a lot of attention by researchers as a key phase in RE. Annie Anton [4] quotes a survey that reveals 40 to 60 percent of software defects or failures to be caused by poor software requirements. Addressing such issues in hindsight may prove to be a costly operation in practice [5], which is why goals have been hailed as a possible remedy with multiple goal analysis methods proposed in the area. In this study we explore KAOS [2], GBRAM [6] and **i*** [7] to get a perspective of how different methods approach goals and what aspects they focus on. We also review other studies comparing the three methods such as those presented by Regev and Wegmann [8] to draw conclusions about the different methods' practical applicability. We also very briefly discuss Evangelia Kavakli's [1] attempt to knit together the different approaches in order to obtain a more rigid RE framework and ask ourselves if such a unified method may have any downsides.

This articles is organised as follows: In Section II we formally introduce goals and explore where they sit in the requirements analysis process. Sections III to V are devoted to examples of goal-based modelling techniques backed by case studies where they are applied in practice. We discuss those techniques in order of formality: from the most formal to the least. In Section VI we weigh up the different methods' strengths and weaknesses and attempt to outline a "recipe to cook up" a new method using the old ones as ingredients. Finally in Section VII we draw conclusions about the felicity of the models presented and summarise our findings.

II. Goals and their Role in RE

Simply put, the goal of a product is the purpose it is intended to serve, such as *execute card payment* for a payment processor. Although we are mainly concerned with goals in software engineering, goal analysis can be applied much more widely as we shall soon convince ourselves. Annie Anton [6] describes goals as *high level objectives of the business, organization or system* that *capture the reasons why a system is needed and guide decisions at various levels within the enterprise*. They are also defined as prescriptive statements of intent whose satisfaction depends upon the co-operation between *agents* and the environment of the software-to-be [3]. What differentiates goals from requirements is that goals explain what should be achieved and why rather than how. Axel van Lamsweerde [2], [3] goes on to classify goals according to different criteria, which we look at a little more closely later in this section.

Van Lamsweerde [2] further defines a *system* as the composite of the software to be produced and its environment. The system is comprised of components, which are divided into passive and active ones, the latter being referred to as *agents*. In a train transport system we recognise the rails as the passive components and the software and humans operating the train system as the active ones. Unlike requirements, a goal may in general involve the co-operation of multiple agents [2]. In the train system example the *safety* goal will typically involve the close co-operation of the on-board train controllers, train tracking software, station system, communication infrastructure, passengers and so forth [2]. One of the important outcomes of the RE process is the decision of which parts will be automated and which will not. Van Lamsweerde stresses that a single agent in the software under development responsible for a goal becomes a requirement, whereas a single agent in the environment becomes an assumption.

A. Goal Classification and Goal Interaction

Goal classification can go a long way towards aiding goal acquisition, refinement, requirements extraction and consistency verification. There are multiple ways of classifying goals and with respect to different factors. We continue following van Lamsweerde [2] to find out how goals can be categorised. Just as ordinary requirements can be broadly divided into functional and non-functional (quality), so too can goals be. A functional goal typically captures some maximal set of desired scenarios; it can be established in a clear-cut sense - the concrete services to be provided. A quality goal, on

the other hand, illustrates desired behaviour when executing functional goals such as reliability, security, usability etc.; these can be less clear. Van Lamsweerde further proposes a detailed subdivision of goals, e.g. satisfaction vs. information goals for functional goals to distinguish between goals concerned with satisfying agent requests and goals that keep agents informed about object states. Similarly, non-functional goals can be divided into accuracy, performance and security goals. Accuracy goals demand the state of the object under scrutiny must be correctly reflected by the system-to-be - violation of these goals may have catastrophic consequences. Performance goals are further grouped into time and space performance goals, whereas security goals into confidentiality, integrity and availability goals.

Temporal behaviour provides a further distinction in goal classification. Achieve (resp. cease) goals stipulate some target property must eventually be satisfied (resp. negated) in some future state. Maintain (resp. avoid) goals restrict the behaviour of the system so that in all states some target property must be satisfied (resp. avoided). Optimise goals favour behaviours that best meet some quality goal. Depending on the level of abstraction, goals can also be high-level (*"serve more passengers"* for a train transport system or *"ensure ubiquitous service"* for an ATM network) or low-level (*"acceleration command delivered on time"* for the above train transportation system or *"withhold card after 3 wrong PIN entries"* for the ATM network) [2].

Separate from goal classification are goal attributes that can be attached to any goal irrespective of its type. The priority of the goal is one such attribute that describes how exigent the satisfaction of that particular goal is and what degree of optionality it can be assigned. These degrees of optionality are, in turn, used as a basis when reconciling conflicting goals. Other relevant attributes may include utility, feasibility etc.

A goal, of course, does not exist on its own. We have mentioned conflicting goals and indeed, a *conflict* link between two goals means that satisfying one goal would lead to the violation of the other. Goals can also *support* other goals and to correctly capture the different links among them, van Lamsweerde uses AND/OR structures. AND-links associate a goal with a set of subgoals each of which must be satisfied in order for the parent goal to hold. Similarly, OR-links require that at least one subgoal should be satisfied for the parent goal to hold. As we mentioned earlier, quality goals may sometimes turn out to be more challenging to express and satisfy in practice. That is why the term goal *satisficing* has been introduced, which eliminates the absoluteness in goal satisfaction and implies goal achievement within tolerable limits.

Besides inter-goal links, there also exist links between goals and other elements of the RE model, such as scenarios. According to van Lamsweerde goals and scenarios have complementary characteristics. The former are general, abstract and make intended properties explicit, whereas the latter are concrete, narrative and leave intended properties implicit. That is why scenarios and goals elegantly complement each other for requirements elicitation and validation.

B. Role of Goals in RE

We continue working with Axel van Lamsweerde [2] to discover why goals are actually of the essence. He cites several reasons, which we discuss below:

- First and foremost goals serve to obtain complete and concrete requirements. They provide a reliable criterion for *sufficient completeness*, which is attained when the requirements specifications extracted can achieve the entire set of goals also taking into account the assumptions made about the environment.
- Pruning irrelevant requirements is also an area goals are widely used. Goals provide a sufficient criterion for requirement *pertinence*.
- Explaining requirements to stakeholders could also be aided by the use of goals, as goals provide the rationale behind requirements. A requirement appears because of some goal that necessitates it. According to van Lamsweerde [2] a goal refinement tree provides the traceability links from high-level strategic objectives to low-level technical requirements.
- Goals provide the right level of abstraction for requirements elaboration and validation.
- Last but not least conflicting requirements can be identified and conflicts resolved with the help of goals.

C. Where to Start Looking for Goals...

Goal identification can pose quite a lot of challenges. As van Lamsweerde [2] writes, they could be directly stated by stakeholders, but also merely implied so separate goal elicitation must be conducted.

Van Lamsweerde also stresses the importance of examining the current (old) system in use the new one is meant to replace. Pinpointing defects in the old system yields a first list of goals of the new system. Once that first stage is complete, these goals must be validated with stakeholders and then refined simply by asking the questions WHY and HOW about the goals readily available. There are also goals that come to light only after a conflict of other goals has been resolved.

Van Lamsweerde rounds off that a goal must be identified and validated as soon as possible to allow for better requirements elaboration. Unfortunately there is no right way to go about obtaining goals and extracting requirements from them, and in the next sections we will see several approaches that share some similarities but also exhibit differences.

III. KAOS

KAOS is one of the best-known software engineering approaches that stresses the importance of explicitly representing and modelling organisation goals [1]. The method stands for "Keep All Objectives Satisfied" and throughout this section we follow Axel van Lamsweerde [5] to get a notion of its structure and practical application. The case study presented in the article discusses a safety injection system for a nuclear power plant and examines the phases from preliminary goal identification to requirements derivation.

A. Initial Goal Identification from the Source Document

After an initial perusal of the documentation of the desired safety injection system, a preliminary set of goals has been extracted. One of those goals is an effective coolant system at all times. The fulfilment of this goal may be hindered by an *obstacle* such as *Ineffective Coolant*, which is mitigated by

the goal *SafetyInjectionIffLossOfCoolant*. This goal is then refined into a subgoal *SafetyInjectionIffLowPressure* and an accuracy property of the environment *SafetyInjectionIffLowWaterPressure*.

B. Formalising Goals

KAOS uses a formal language of symbols to ensure correctness and completeness of goals and to aid the requirements elaboration process. In addition to the usual logical concepts and symbols (e.g. "$\rightarrow$" - implication and "$\leftrightarrow$" - equivalence) KAOS introduces several temporal operators explained below:

$\Diamond P$	P holds in some future state
$\Box P$	P holds in all future states
$A \Rightarrow C$	In every future state A implies C i.e., $\Box (A \rightarrow C)$
$A \Leftrightarrow C$	In every future state A implies C i.e., $\Box (A \leftrightarrow C)$
$\bullet P$	P holds in the previous state
$@P$	P has just become true i.e., $\bullet \neg P \wedge P$

Here is an example showing the formal definition of the goal *SafetyInjectionIffLowWaterPressure* [5]:

Goal: maintain *SafetyInjectionIffLowWaterPressure*.
Informal definition: a safety injection alert should be triggered if and only if the water pressure is below some critical threshold.
Formal definition: *TriggerSafetyInjectionAlert* $\Leftrightarrow$ *WaterPressure* $<$ *'Threshold'*.

C. Detecting and Resolving Conflicts

Requirements engineers live in a world where conflicts are the rule rather than the exception [2]. They arise from diverging viewpoints and concerns. Conflicts must be rooted out as early as possible and goals can play a vital role in that process. In the case study under consideration [5] a safety injection alert must be avoided during normal start-up or cool-down.

Goal: avoid *SafetyInjectionDuringNStartup/CoolDown*.
Informal definition: a safety injection alert should not be triggered during start-up/cool-down even if the water pressure drops below the critical threshold.
Formal definition: $(NStartUp \vee NCoolDown) \Rightarrow$
$\neg$ *TriggerSafetyInjectionAlert*.

As can be easily established, in the previous subsection the goal to maintain *SafetyInjectionIffLowWaterPressure* is not at odds with the goal we just defined, unless the power plant happens to be in a start-up or cool-down state when the water pressure sinks below the threshold. The conflict resolution tactics we can employ [5] lead to the weakening of the goal of maintaining *SafetyInjectionIffLowWaterPressure*, which, in turn, becomes:

Goal: maintain
SafetyInjectionIffLowWaterPressureUnlessStartUp/CoolDown
Informal definition: a safety injection alert should be triggered whenever the water pressure is below some critical threshold, except during normal start-up and cool-down.
Formal definition:
$\neg(NStartUp \vee NCoolDown) \wedge$ *WaterPressure* $<$ *'Threshold'* $\Leftrightarrow$ *TriggerSafetyInjectionAlert*

D. Refining Goals and Assigning Agent Responsibilities

As we pointed out in section II, a requirement is a single agent responsible for one goal. Ultimately, we strive to extract concrete requirements from the goal analysis, ergo we must reduce each goal to an objective within the responsibility of one agent. Such a goal would require the safety injection alert system to constantly monitor the current operating regime [5], which is impracticable and therefore the goal to maintain *SafetyInjectionIffLowWaterPressure* is unrealisable. To resolve the situation Letier and van Lamsweerde employ an agent-based refinement tactic and split the goal into two subgoals, also introducing a state variable *Overridden* that monitors the current operational mode of the power plant. Formally the two subgoals are defined as follows:

Subgoal: maintain
SafetyInjectionIffLowWaterPressureUnlessOverridden
Formal definition: *TriggerSafetyInjectionAlert* $\Leftrightarrow$
WaterPressure $<$ *'Threshold'* $\wedge \neg$ *Overridden*.

Subgoal: maintain *Overridden*
Informal definition: the state variable *Overridden* contains information about the current working regime of the power plant. It is **true** if the plant is in a state of normal start-up or normal cool-down and **false** otherwise.
Formal definition: *Overridden* $\Leftrightarrow (NStartUp \vee NCoolDown)$.

Letier and van Lamsweerde go on to claim that a human *Operator* agent may be placed in charge of monitoring the current operating regime of the power plant and thus this agent can directly control the state variable *Overridden*. That is how both subgoals become realisable. The role of the human agent becomes an *environment assumption*, whereas the automated safety injection alert becomes a *software requirement*.

E. Predicting and Resolving Obstacles

The first round of goal identification tends to be overidealistic and to overlook possible unexpected behaviour [5]. Not anticipating exceptional behaviour may result in incomplete and unrealistic requirements. An *obstacle* is such an exceptional scenario that may hinder goal satisfaction. Formally an obstacle O is said to obstruct a goal G if and only if:

$$\{O, Dom\} \models \neg G \text{ obstruction}$$

$$Dom \not\models \neg O \text{ domain consistency,}$$

where Dom denotes the software environment and its assumptions.
Obstacle analysis adopts a pessimistic view of goals, requirements and the software environment. It looks for ways to violate the assumed behaviour of the agents to yield a more robust and less error-prone system.
In the security injection system example Letier and van Lamsweerde cite a failure in the activity of the human agent as a possible obstacle. Under normal circumstances we would assume that the human *Operator* agent accurately updates the value of the state variable *Overridden* whenever the power plant switches operational mode between normal start-up and

normal operational mode, or between normal start-up and normal cool-down. More formally, our assumption states the following:

Assumption: avoid *OverriddenWhenNoNStartUp/NCoolDown*
Informal definition: outside normal start-up and cool-down the state variable *Overridden* should be set to **false**.
Formal definition: $\neg@(NStartUp \vee NCoolDown) \Rightarrow \neg@Overridden.$
ResponsibilityOf: *Operator*

Let us now imagine the *Operator* agent accidentally activates the state variable *Overridden* after the power plant has exited its normal start-up mode and entered regular operational mode. We must stress that we completely rule out any malicious intention or grievous negligence on the part of the agent, i.e. that they would deliberately or inadvertently disable the security injection system at a moment when it may be needed. This yields the following obstacle:

Obstacle: *Operator* fails to update *Overridden*
Informal definition: the *Operator* inadvertently overrides the security injection system while the power plant is in regular operational mode and the water presure is still above the minimum.
Formal definition: $\Diamond(\neg@(NStartUp \vee NCoolDown) \wedge @Overridden)$

Once envisaged, obstacles must be adequately addressed. Obstacle resolution techniques involve assessing the criticality and the likelihood of the obstacle, exploring alternatives and selecting the best one based on a complex of factors such as cost, practicability, risks, performance, etc [5]. The workaround suggested by Letier and van Lamsweerde [5] involves *strengthening* the responsibility of the automated security injection system to include monitoring the water pressure at all times. Thus we obtain another goal for the automated system:

Goal: maintain *Overridden* depending on water pressure
Informal definition: the automated safety injection system should at all times monitor the water pressure and leave **true** as the value of the state variable *Overridden* if and only if it was already set to **true** by the human *Operator* in the previous state and the water pressure had dropped below the threshold.
Formal definition: $@Overridden \Leftrightarrow @Overridden \wedge$
$\bullet($ *WaterPressure* $<$ *'Threshold'*$) \wedge$ *WaterPressure* $<$ *'Threshold'*
ResponsibilityOf: automated security injection system

Obstacle analysis is an iterative process [5]: the solution of one obstacle may give rise to others that need addressing. After strengthening the responsibility of the automated security injection system one may ask what the purpose of the human agent is and if it is cost-conscious to employ an error-prone human only to monitor when the power plant switches working regimes. Obstacle analysis cycles involve trade-offs among various non-functional application-specific goals regarding safety, reliability, cost, performance, etc. It is a constructive method particularly relevant to high assurance systems as many deficiencies are known to be caused by incomplete risk assessment and poor design not properly appraising the human factors and other sources of failure.

F. Extracting Operational Requirements from System Goals

Having resolved the obstacle in the previous subsection we can now proceed towards turning that goal into a requirement. The transition presented here is simple and straightforward and uses the so called 'Immediate Achieve' operationalisation pattern [5]:

Operation: *OverrideSafetyInjection*
PerformedBy: automated security injection system
Input: *Overridden, WaterPressure*; **Output:** *Overridden*
DomPre: *@Overridden,* $\bullet($ *WaterPressure* $<$ *'Threshold'*$) \wedge$ *WaterPressure* $<$ *'Threshold'*;
DomPost: *Overridden*;
ReqPre/TrigFor: *SafetyInjectionIffLowWaterPressureUnlessOverridden*

The table above states that the automated security injection system will execute a check every time the state variable *Overridden* has been switched on by the human *Operator* agent and will set *Overridden* back to **false**, unless the water pressure was already below the threshold when the *Operator* enabled *Overridden*. The **ReqPre/TrigFor** keyword indicates the result of this operation will contribute to achieving the goal mentioned.
Letier and van Lamsweerde conclude the section by pointing out that goal-oriented requirements elaboration ends where most traditional requirements specification techniques should start. They highlight the difference with use-case driven modelling, which leaves goal achievement implicit, tends to focus on too narrow an area thereby "missing the point". The goal-oriented requirements elaboration approach employed here, in contrast, started at a higher-level, constantly keeping goal achievement in mind.
Having explored the inner workings of KAOS, we can now draw some conclusions about the method in general. It is undoubtedly very formal and one may rightfully suppose it is most suitable for large critical systems such as the safety injection one discussed here. We would normally associate such systems with the waterfall model - a conservative model that aims to resolve all issues in the current phase before proceeding to the other.
In the next section we explore a slightly less formal method, which, despite lacking its own language, also takes a rather conservative stance on goals.

IV. GBRAM

In this section we turn our attention to another method that places goals in the centre of RE - the Goal Based Requirements Analysis Method (GBRAM) due to Annie Anton. Throughout the section we will be following Anton [6] to gain insights into the method's elements and look into a case study conducted to demonstrate its applicability. GBRAM was put into practice when analysing the Career Track Training System (CTTS), part of a business re-engineering project for an Air Force Base (AFB). This was not a software project per se, but a wider scheme aimed at promoting career development. Nevertheless goal analysis proves to be very relevant as we shall convince ourselves below.
As we have already seen, goals can be very useful in elaborating requirements. In general, GBRAM focuses on two important aspects. The first is a question related to goal

analysis and is by nature similar to the goal analysis in KAOS *How are goals identified?*. The second aspect is, a little misleadingly, posed as goal evolution, but in fact it addresses the change of goals during the goal analysis and refinement process rather than stakeholders' intentions making a sudden U-turn.

A. Goal Analysis

The initial goal deduction is an intricate process of extracting information from process descriptions such as flow charts, entity Relationship (ER) diagrams, etc. Furthermore the practitioner should look for additional sources such as interviews with stakeholders. Anton points out that stakeholders tend to express their requirements as actions and operations rather than explicit goals. Thus the analyst must look for verbs and other words expressing action to infer a stakeholder's goal. In her investigation of the CTTS Anton distinguishes between achieve and maintain goals. Achieve goals answer the questions *Is completion of this goal self-contained?*, *Does this goal denote a state that has been achieved or a desired state?* and *Does achievement of this goal depend on the completion of another goal?*. Maintain goals on the other hand are recognised by asking *Does this goal ensure some condition is held true throughout all other goal operationalisations* and *Is continuous achievement of this goal required?*. Such a goal is G_2 in table I, which must be achieved on a continuous basis. Maintain goals can also be identified by searching for words such as 'supply', 'provide' and any other verbs suggesting a continual state within the system.
After classifying goals into achieve and maintain goals (see tables I and II), Anton directs her attention to the identification of agents and stakeholders. She defines the agent as the entity responsible for achieving a given goal. Stakeholders on the other hand are the parties directly or indirectly concerned by the achievement or failure of the goal. Tables I and II below are an excerpt of the results Anton obtained after a thorough investigation of the CTTS documentation.

Maintain Goals	**Agent**	**Stakeholders**
G_1: Career tracks provided	AFB	AFB
G_2: Tax payers money spent efficiently	AFB	AFB, employee
G_3: Training co-ordinated	AFB	AFB, employee

TABLE I: Maintain Goals, Agents and Stakeholders

Achieve Goals	**Agent**	**Stakeholders**
G_4: Career portfolio created	employee	employee, supervisor
G_5: Course completed	employee	employee
G_6: Skills improved	employee	AFB, employee
G_7: Certification granted	AFB	AFB, employee

TABLE II: Achieve Goals, Agents and Stakeholders

Let us now briefly comment on some of the goals, agents and stakeholders identified above. Goal G_1 : *Career tracks provided* in table I, for example, has been identified as the sole responsibility of AFB, which is also the goal's only stakeholder. The reason for this according to Anton is that employees are responsible for and interested in their *individual* career development rather than the opportunities provided to all. Moving on to table II we examine G_6 : *skills improved* and observe that although each employee is in charge of their own skills, the party interested is also the employer, who can only benefit from highly-skilled workforce.

B. Goal Evolution

As Anton points out, objectives are very likely to change in time, for example because stakeholders reconsider their priorities and re-define their goals. She mentions two types of goal evolution: elaboration and refinement, although she does not make the distinction between the two very clear.
Goal elaboration is attained through very useful techniques such as identifying obstacles and analysing scenarios and constraints. Just as in KAOS, GBRAM also considers the question how the execution of a goal may be hindered, in order to anticipate exception cases. For example, an obstacle to G_7 : *Certification granted* may be that the employee does not obtain her certification after all. Scenarios help concretise ways in which a goal can fail. In our current example this may be that the employee fails the training course or does not attend it at all. Another benefit of scenarios is that they help discover hidden goals. Anton remarks that the question *What happens if...?* leads to new relevant goals that may have been overlooked during the initial analysis.
Goal refinement is realised by reconciling synonymous goals, eliminating redundancies or establishing precedences among goals. Anton also considers identifying goal constraints to be a refinement as well as an elaboration technique. A constraint is different from an obstacle in that it does not completely prevent a goal from being achieved, rather imposes further conditions that must be met to fulfil the goal. For instance a constraint related to goal G_7 : *Certification granted* may be that certification should enable employees to advance to another level.
Another technique that aids goal refinement is establishing a precedence relation in the set of goals, i.e. which goal needs to have been completed for another goal to follow. The question we can ask ourselves is *Which goals are prerequisites for this goal?* Examining table II we can see that for instance, goal G_5 : *Course completed* must be a prerequisite for goal G_7 : *Certification granted*. This is denoted $G_5 < G_7$. Goal precedence enables us to envisage goal operationalisation and consider further refinements and elaborations.

C. Goal Operationalisation

Turning goals into requirements is what we would ultimately like to achieve. To that end Anton proposes consolidating goals into goal schemas that specify the relationships between goals and their agents. Goals and their operationalisations (actions) are specified as events in terms of preconditions and postconditions. A precondition must be met for the goal to be achieved and a postcondition is the state of the system after the goal is completed. A precondition for certification to be granted can be, for instance, that the employee has passed the certification examination. A suitable postcondition after the certification has been granted may be that the employee

progresses to a higher level. A formal goal schema similar to the one in Anton's article [6] is presented below:

Goal:	Certification granted
Type:	Achieve
Description:	AFB must award certification to employees who have successfully completed the appropriate training and passed the examination.
Action:	Confer certification
Agent:	AFB
Stakeholders:	AFB, employee
Obstacles:	Certification not granted
Scenarios:	Employee does not attend training. Employee does not pass final examination.
Postcondition:	Employee advances to a higher level.

TABLE III: Goal schema for goal G_7

Anton concludes by remarking that the goal schema above is what translates into a requirement. By elaborating and refining each goal in the set of goals we can thus arrive at a set of requirements.

V. The i^* Framework

The i^* framework takes a slightly different approach from what we have seen thus far. While it still stresses the importance of goals in RE, it delves even more deeply to the very core of the underlying organisation to find out why the new system is needed and where it sits in its surrounding environment. The i^* framework consists of two modelling components: Strategic Dependency (SD) used to discover dependency relationships among various actors in the organisation; and Strategic Rationale (SR), which describes stakeholder interests and concerns and how they may be addressed by various configurations of system environments. Throughout this section our constant companion will be Eric Yu [7], whose work on the i^* framework is illustrated in a case study focusing on a computer-based meeting scheduler. Yu considers the fundamental building block of the i^* framework to be the intentional actor. Actors are assumed to have intentional properties such as goals, beliefs, abilities and commitments. Actors may depend on and collaborate with each other to attain a goal that may be impossible for one actor to attain single-handedly.

A. Strategic Dependency

The formal requirements of the computer-based meeting scheduler we are considering in this section might state that an initiator invites all participants to enter the times and venues convenient for them and those that are impossible for them, and the scheduler should produce a solution suitable for all parties. Yu points out there have been multiple methodologies developed to refine such kind of requirements to achieve more precision, completeness and consistency, but to develop systems that truly meet an organisation's demands, one needs to gain a deeper understanding of how the system fits into the organisation. The questions that need addressing first focus on "*why*" rather than "*what*" and "*how*"[7]. These are questions such as "*Why is it necessary to schedule meetings ahead of time?*", "*Why does the initiator ask about convenient and inconvenient timings?*" and "*Are some participants more important than others?*".
Thus the Strategic Dependency model clarifies the intentional relationships among actors. In the meeting scheduler example by asking the above mentioned questions and similar we can deduce that the meeting initiator depends on the participants to enter the timings convenient for them and subsequently attend the meeting on the agreed date. Yu also singles out important participants as constituting a critical dependency in the meeting scheduling task. Yu stresses that establishing these dependencies is critical not only for the successful development of software systems, but also for the collaboration with other software systems as well as the accommodation of their ongoing evolution.

B. Strategic Rationale

The Strategic Dependency model provides a framework for establishing external relationships among actors. Strategic Rationale goes one step further and probes into the actors themselves, their tasks, goals and resources and the internal relationships therein. In the meeting scheduler example the initiator has one clear goal: to arrange the meeting. However this goal can be split into tasks, subgoals and obtaining resources. One such task may be to obtain available dates from the participants and based on that resource work out a time suitable to all invited participants. As Yu remarks, a soft subgoal for the initiator may be that the tasks should be executed quickly or with low effort, which is where automation and the computer-based scheduler come in. The initiator can delegate the task of polling participants' opinions to the computer-based scheduler. SR thus provides a way of modelling stakeholders' interests and how they can be met.

C. Supporting early-phase RE

Unlike RE, which traditionally aims to identify and eliminate incompleteness, goal analysis with i^* focuses on establishing user goals, their relationships and the possible alternative ways in which they can be satisfied. In order to weigh and compromise between different alternatives, i^* offers a number of levels of analysis: *ability*, *workability*, *viability*, *believability*.
When an actor is allowed to perform a certain action or execute a procedure, Yu says that the actor is *able*. Different actors may have different permissions to perform different actions. Ability to achieve a goal requires the satisfaction of all subgoals and completion of all subtasks associated with it. In the meeting scheduler example the initiator was able to organise the meeting if she managed to obtain sufficient information about availability from the invited participants and work out a date suitable for all.
An action is *workable* if and only if there exists a means of achieving it [7]. The initiator of the meeting could, for example, consider the intersection of all the appropriate timings submitted by the invited participants.
A workable solution may not necessarily be *viable*. Computing

the intersection of all available time slots is laborious and error-prone. The soft goals that stipulate the action should be completed quickly and with minimum effort will fail to be satisficed.
Judging whether a proposed solution is viable and workable is based on assumptions that fall into the category of *believability*. The believability of a rationale can be verified by considering each of its assumptions. For example the workability of the solution involving the intersection may be justified by years of experience employing the method.
Yu concludes that these four criteria can go a long way towards refining objectives and elaborating goals. Although requirements extraction from goals is not as evident as in the previous two methods, i^* provides the first step towards reasonable RE.

VI. Comparisons and Unifying Models

In this section we pause and look back on the methods we have familiarised ourselves with in the previous sections and try to find similarities and differences led by Regev and Wegmann [8]. We also briefly study a model metamethod due to Evangelia Kavakli [1], that is, a a procedure to splice approaches together.
We start by discussing the common features of the methods. Surely, they all try to understand why a new product is being developed, but i^* seems to go that extra mile and delve even more deeply into each and every actor's priorities to establish the dependency and precedence relationships among the goals. *Stakeholders* are recognised by all three methods as the parties interested in attaining the goal. And while KAOS aspires to assign one goal per agent to derive requirements, i^* is perfectly happy with actor co-operation and does not try to refine goals any further.
A significant difference is that KAOS and GBRAM study goals with a view to extracting requirements - they are intended for completion *before* the formal RE begins, whereas i^* does not perceive itself as an independent stage of the software life cycle. Its purpose is just to aid early RE and does not constitute its own phase. Nor does it directly lead to requirements extraction. Although KAOS and GBRAM share many similarities, it is interesting to observe that KAOS does not consider goal constraints. GBRAM, on the other hand, does not bargain for conflicting goals. Also, while KAOS introduces a formal language to define and analyse goals, GBRAM remains more verbal and abstract and does not formalise its procedures. Table IV features the main aspects of the methods and is adapted from a similar table presented in [8]:

	KAOS	GBRAM	i^*
Goal	a nonoperational objective to be achieved by the composite system	targets for achievement which provide a framework for the desired system	not formally defined
Achieve Goal	a property that holds in the current or come future states	objectives of an enterprise or system [...] satisfied when the target condition is attained	not used
Maintain Goal	property that holds in the current and all future states	goals which are satisfied while their target condition remains constant or true	not used
Softgoal	not used	not used	a qualitative criterion that must be met when satisfying other goals
Constraint	not used	a condition on the achievement of a goal	implicit in the SD model
Obstacle	exceptional behaviour that may result in incomplete or unrealistic requirements	behaviours or other goals that block the achievement of a given goal	implicit in the SD model

TABLE IV: Overview of goal concept definitions in KAOS, GBRAM and i^*

Now that we have weighed the pros and cons of the models we explored so far, it would be interesting to see if there is a way we can combine their strengths to use them to our advantage. Evangelia Kavakli [1] proposes a framework of four knowledge modelling states that could be useful in goal analysis and subsequent RE:

- The *As_Is* state concerns knowledge about the current organisational situation.
- The *Change* state refers to the reasons (the need) for altering the existing situation.
- The *To_Be* state characterises knowledge about the future enterprise situation.
- The *Evaluation* state concerns knowledge regarding the assessment of the current situation, the suitability of a change plan, or the appropriateness of a future enterprise model.

It would thus be well-advised to choose the best method that fits each state from the three we have reviewed in this study. Based on what we have learnt, the *As_Is* and *Change* states are best addressed by i^*, because they are more general. The Strategic Dependency model of i^* deals exactly with the present situation in the organisation and the need for change. A combination of KAOS and GBRAM with all the aspects considered would best portray the software to be, also not forgetting goal validation both methods promote.
A question, of course, arises whether the formality of KAOS is in fact compatible with the generality of i^*. Indeed the waterfall-like nature of the former may sometimes appear to

be miles away from the almost agile characteristics of the latter. In the author's opinion, however KAOS and i^* really do complement each other. Recalling the first stage of KAOS, we can easily convince ourselves that initial goal identification can only be aided by studying the underlining organisation. We can thus conclude that those two methods do manage to co-exist to yield a stronger, more rigid procedure.

VII. Summary and Conclusions

In this study we explored the values of goals and investigated where they sit in the software development process. We also familiarised ourselves with different approaches designed to analyse and use them for the extraction of requirements. Finally we drew parallels among the three models and briefly discussed how we could make the most of all three in one unified model.

References

[1] E. Kavakli, "Goal oriented requirements engineering: a unifying framework," *Requirements Engineering Journal, Springer-Verlag London*, vol. 6, pp. 237–251, 2002.

[2] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, ser. RE '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 249–. [Online]. Available: http://dl.acm.org/citation.cfm?id=882477.883624

[3] ——, "Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]," in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, Sept 2004, pp. 4–7.

[4] A. Anton, "Successful software projects need requirements planning," *IEEE Software*, vol. 20(3), pp. 44,46–47, 2003.

[5] A. van Lamsweerde and E. Letier, "From object orientation to goal orientation: A paradigm shift for requirements engineering," in *Radical Innovations of Software & System Engineering, Montery02 Workshop, Venice(Italy), LNCS*. Springer-Verlag, 2003, pp. 4–8.

[6] A. I. Anton, "Goal-based requirements analysis," in *Proceedings of the 2Nd International Conference on Requirements Engineering (ICRE '96)*, ser. ICRE '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 136–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850944.853130

[7] E. S. K. Yu, "Towards modeling and reasoning support for early-phase requirements engineering," in *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, ser. RE '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 226–. [Online]. Available: http://dl.acm.org/citation.cfm?id=827255.827807

[8] G. Regev and A. Wegmann, "A.: Where do goals come from: the underlying principles of goal-oriented requirements engineering," in *In: RE 2005. Proceedings of the International Requirements Enginering Conference (2005*, 2005, pp. 253–362.

User Stories and Business Impact

Qian Zhou
Department of Computer Science
University of Helsinki
Gustaf Hllstrmin katu 2b, FI-00014
Email: qxzhou@cs.helsinki.fi

***Abstract*—There are two fundamental roles in any type of business transaction. One role is a customer who wants or needs something and the other one is the provider who can offer needs. However, there is gap between what the provider offers and what the customer needs. Fast release of a software product based on marketing demand has more opportunities to conquer a market without too much competitiveness. How to develop a product that meets customers' needs and releases it in time are essential for software companies. Software development project is about to develop features which are needed by users. In agile software development, user stories are used to describe features and functionality of a product from the perspective of a user. As a result, user stories identify user needs and stand for units of business value delivered by the software company. User stories also need to be continuously revised according to change of user needs in the development evolution. In order to develop right product rightly and adjust project requirements based on user needs, it is worth to research how user stories could affect a business and how to use user stories correctly and efficiently.**

Keywords*—Software Engineering, user stories, business value delivery, impact mapping, agile development.

I. Introduction

A product can be described by a list of features. It is the feature that makes the product valuable in marketplace and motivates customers to purchase a product. One aim of a software development project is to build features that are valuable to the customers. However, not all software projects could delivery business value correctly. Jim Johnson and his Standish group found that only 20% of the features in a product is always or often used while 80% of the features is seldom or never used[10].

The best way to build a software product that meets users' needs is to begin with user stories[2], which is an agile software development methodology.Users stories are brief statements that describe a product. User stories are also tokens that stimulate the discussion process , via which the design solutions could be explored and discovered. Finally those solutions could be implemented in the later development process. There are several user story formats and one template introduced by Mike Cohn is" As a [user], I want [goal], so that[reason]"[2]. Each user story is expected to make an contribution to the value of the overall product once it is implemented. As a result, user stories could help the product development team to identify user needs and to implement valuable features. It also helps the team to avoid wasting time and money on unneeded features.

There is no doubt that how to develop a right product rightly is a topic worthy to study. In this paper, user story and its business impact from different aspects will be discussed and tips to use user stories will also be covered. Section 2 will give an overview of user stories and the content includes 3C components, INVEST Model, alternatives of user stories as well as benefits and limitations of user stories. Section 3 will discuss business impacts of user stories. Section 4 covers methods on using user stories. Finally a summary on user stories will be given.

II. Overview of User Stories

This section will cover user stories details such as 3C components and INVEST model as well as its benefits and limitations.

User story was firstly mentioned by Kent Beck in the Extreme programming development method. Originally Extreme Programming described a user story as a small amount of text written on an index card to function as a reminder for a conversation between the developers and the customers[1]. There are different templates for the user story, one example proposed by Dan North is "As a X, I want Y, so that Z"[22]. X here is the role form the point view of a user, Y is goal or achievement and Z could be the benefit that users get or business value delivered by the team. A detailed example could be "As a registered user, I want to log in, so i can access content that is only accessible for subscribers." Mike Cohn suggests that the value is optional, which means that for a story template such as As a [type of user], I want [some goal], so that [some reason], the "so that" clause is optional[21]. However, other people suggest the opposite format which focuses on value. The suggested format is "In order to [achieve some value], as a [type of user], I want [some functionality]"[7]. In general, as long as the user story acts as a reminder for discussion, those form are usable.

A. Ron Jeffries' 3C

Card, Conversation and Confirmation[3] are three components proposed by Ron Jeffries for a user story. User stories are written in physical form : an index card or a sticky note.

Cards are tokens that identify requirements and briefly remind what value of the story is going to be delivered[3]. The card has story title and brief description which summarizes the story content. The card could be used for project planning via initially estimating effort such as time and technical skills to implement a story. The card also plays an important role in reminding future conversations between the develop team and users.

Conversation covers all the details of a story via exchanging ideas or opinions between the customers and the develop team[3]. The conversation is mainly done by verbal but could also be handled with documentation or recording videos for the conversations. It is recommended to focus verbal conversation, as documentation costs long time than verbal communication and also puts burden on the single person who documents. When a bunch of documents replies on that person to handle, a bottleneck of the development process could be caused[5]. During the conversation phase, detailed user requirements can be clarified and solutions can be framed for the story implementation in the development process.

Confirmation of user stories is done by acceptance test from users with the purpose to show whether the user stories could be implemented correctly[3]. Acceptance test also could be called customer test or functional test, which tells the develop team whether the system acts in the way the customer expects. Except confirmation from the users side, developer team also needs to confirm that the acceptance test can pass or run successfully to show the story is successful. This lets developer to know they have satisfied the user needs to build the"right" software.

B. INVEST Model

Bill Wake proposed the INVEST(Independent Negotiable Valuable Estimable Small Testable) model to cover characteristics of a good story[4].

Good stories should be independent to each other so that they can be implemented and scheduled with any order. This is because dependencies among stories could limit flexibility of moving the stories around when the product owner prioritize the user stories in backlog.

A good story should be negotiable for its details and changes. As the story card just briefly describe the story without details. The details come out from the conversation phase between develop team and users. User needs may change during the process of development or there is a need to change the user stories. Negotiability enables teams to discuss story details, discover new options and make changes for the stories when needed.

The first principle[15] of the Agile Manifesto points out to deliver valuable software to the customers. Each story is a small and independent behavior that presents some value to the user. A good story should be valuable to the users since users are the ones who will pay money for features of the product. This characteristic requires the team to develop needed features and delivery right business value. Value of the stories can be used to decide the prioritized order in the backlog and success of a business depends on the value of the team can deliver.

A good story can be estimable, this means that the skills and time to be spent for implementing the user stories can be roughly estimate. This not only helps the customers to prioritize the story's implementation order but also requires the develop team to understand the stories correctly and what skills need are needed to implement the desired features.

The size of the user story should be small, so the development team could know the story scope and get more accurate estimates. A story with big scope is hard to understand and estimate.

A good story should be testable. This requires the team know what they are going to implement and could write the test for the story. Confirmation of the tests also helps the team to know whether their is met or not.[4]

C. User stories and requirements

A requirement is a capability that a product outcome should conform. The requirements of a product should be documented, tested and measured to identify business needs or opportunities. Use stories are kind of requirements but they are also different from requirements.

User stories can be updated when needed while requirements documentations are usually out of date. User stories encourage face-to-face conversation and based on latest leanings. While requirements encourage guesswork or assumption and the team members have little updated learning. User stories require participation of end users while there is no such need in the documented requirements. In addition, user stories are written in non-technical and understandable way so the users can write the user stories. However, requirements are usually documented in technical and complex way.

User stories enables real-time and quick feedback, which are disabled in requirements. User stories allow team work while requirements discourage open collaboration. User stories are relatively easy to estimate effort to implement, so they simplify the planning while requirements make the plan more complex. Overall, it is easy, fast and cheap to create user stories while it is hard, slow and expensive to write the documents.

D. User stories and use cases

Use cases consist of a list of steps which define how a role interact with the system. While user stories are short description of features.

There are some common points between use case and user stories. They both are techniques that used to capture requirements. They both put users at the center of the development effort and both need to be verified by test cases.

However, there are differences between use case and user stories. Use cases include different structured scenarios such as main success scenario, alternative scenarios and failure scenarios. Use cases describe how the users interacts with the product by using one or more scenarios. While user stories describe how the user employs the product.

Both use cases and user stories are sized to deliver business value but they have different scope. User stories have smaller scope as the size of story is constrained to schedule work. However, use cases covers larger scope than the story. User stories can be used for planning as difficulty and time to implement the story can be estimated in user story. While use cases are too large to offer useful estimate. Use cases and user stories differ in their longevity. Use cases exist along the product development process while user stories only exist each iteration in which they are added to the software.[20]

E. Benefits and Limitation of User Stories

When compared with use cases and traditional requirements, user stories have the following advantages.

Firstly, user stories are more precise due to emphasis of verbal communication[20]. Compared with written requirements, face-to-face communication between development team and users could overcome inaccuracies in written language and also improve close relationship with users. In return, close relationship also results more effective knowledge transfer among team and users, who could actively participate in designing of the system[9].

Secondly, user stories can be used for project planning as each user story has estimation on level of difficulty and time consumed to implement it[20]. User stories could also be used for scheduling, this means that user stories can be assigned with priority order.

Thirdly, user stories allows team to defer details. A story could be replaced with more detailed story when needed[20]. This also means that user stories are suitable for iterative development that allows stories to be written and revised when needed[9]. User stories are flexible and agile as they allow team to add, modify and delete user stories during the development process. In addition, user stories help the team to know what is going to build and make goals more clear. It also helps to reduce upfront planning time.

Despite the advantages of user stories, there are also some limitations. One limitation of user stories is that different people could have different interpretations for a user story due to its simplicity. As a result, conversation is needed to clear any doubts in face-to-face communication. In addition, simplicity of user stories also causes to ignore requirements in details. It is known that when creating user stories, user stories focus on roles. However, William Hudson argued that roles could not express needs and behaviors of users in a full picture. A user story describes activities the user may take but have no clue about when and how a task should be performed[8]. He suggested to use persona stories.

III. User stories and business impacts

This section will discuss user stories' affect towards different aspects that involve with business.

A. User stories could clarify aim

A business aim is what the a business wants to achieve. A primary aim for all business companies is to add value to their product which could make a profit. However, at the early stage of business, it is hard to identify what kind of product features should be delivered to the customers. Without a clear picture on what is going to be built, it is hard to align business aims and development actions or effort.

User stories can help the development team to find their aim and then clarify the aim. User stories are good and easy start to explore possible features to be built with participation of the users.

User stories act as reminder for conversation between the development team and the users. During the conversation, user needs can be discussed in details and aim on meeting those needs can be more clear. Solutions and design ideas can be also framed in the conversation phase, techniques needed for those solutions can be carefully chosen to meet the aims. Continuously applying and refining user stories as routine and getting user feedback during the development process help the team to better adjust and target the direction of aim and then build product that is valuable in the marketplace.

In addition, as user stories must be verified by the users. By obtaining the agreement from users and successful pass from the acceptance test, the development team can ensure their aim on building needed product is right.

With a good and clear aim of building what valuable product to the customers, software company could have more chance to be successful.

B. User stories could reduce risk

User stories can yield value by describing valuable features that users are going to buy for the product. But user stories' value can be more than on the side of the end users.

User stories could reduce risks. There are many types of risks in a business, for example, financial risks, strategic risks and technical risks. Financial risks involves with amount of money and time to be spent on developing the product. Strategic risks could be highly tight with the business plan on when to release a product with less competitors. Technical risks could be skills and resources needed to implement a story.

As user stories can be used to estimate time and effort to implement a story, this could also help the team to have a rough idea on when they can release a product.

Technical risks can be discovered during the early stage of product development by applying user stories. User stories need to be discussed, during the conversation, detailed technical skills to implement the story can be specified. Technical risks can be also discovered when compared the needed skills and what skills the development team has. Then solutions on avoiding risks and approaches on solving those technical problems could be further discussed and found.

In addition, during a group discussion, every participate is a "risk detector". When one member proposes a solution, the others could evaluate that idea and detect possible risks within it. Then a quick respond towards this possible risk could be exposed for team discussion. This reduces the chances to risks and ensure quality of the product in some degree.

What is more, it is also easier to fix problems caused by some technical risks in the earlier stage than the later phase.

As user stories must be verified and agreed by the users before implementation. The acceptance criteria phase also reduces the chances of delivering unneeded features to the marketplace.

C. User stories can help to reduce cost

One benefit of user stories is that they can reduce cost caused by changes. Changes during a product development process are unavoidable. With iterative development, modifications of user stories are allowed at each sprint. The earlier the changes are made and the less cost paid for the change

will be needed. User stories are allowed to be modified based on user feedback and development needs. Quick user feedback could let developers implement the latest user needs and update features in early stage before more complex features are implemented on the top of old version feature. User participation and quick respond not only save implementation time but also reduce complexity for developers to refine features.

D. User stories can stimulate innovation

Innovation is one thing that not only adds value for a business but also make that business less competitive in a marketplace. User stories can bring innovation for a business due to following reasons.

As known, user stories are short descriptions of product features. Some people argue that user stories' brief causes uncertainty and ambiguity, which could lead different versions of understanding for the same stories. However, this could be positively viewed as a good thing[23]. When different team members interpret the stories differently, it also means different prospective and ideas. What the ambiguity causes is that the team takes relatively longer time to discuss each different interpretations and reasons. The teams always need to discuss, it is more likely to be worth on spending longer time to find a potential innovative idea rather than follow normal ideas with less time.

When apply user stories, conversation also means exchange ideas among the team. This also means that everyone is learner but also knowledge spreader. People could learn from each other during the conversation and their knowledge could also be updated at the same time. Updated learning in turn could stimulate innovative ideas when combine specific previous background. Group discussion among different groups such as developers and customers also means combination of different perspectives, which flourishes innovation.

IV. Tips on using user stories

In agile manifesto, a user plays an important role in making decisions on what is going to be built [15]. XP (Extreme Programming) encourages customers to write user stories and then discuss details with the team members directly[1]. This section will cover tips on improving user stories based on a book called 50 Quick Ideas to Improve Your User Stories[5]. User stories creation, discussion, planning and prioritization will be covered.

A. Stories creation

User stories are assumptions of business value, creating valuable stories is the very first step to be successful in business. There are following tips for creating stories based on.

Firstly when creating user stories, it is suggested to avoiding documenting details of the user stories[5]. Documenting detailed stories costs time and causes bottleneck or failure to rapid change due to single person documentation[5]. There are cases that need to write down the details of stories. For example, it is good to write details of stories that needs specialist knowledge from a person who does not participate the discussion.

Secondly, when creating user stories, Gojko suggests to not to pay too much attention to the story format as long as the key elements are included. One function of the story card is a reminder for discussion. Once a story format stimulates a good discussion and it will serve the purpose. Any format of the story is good to start the discussion when the information on the card is true. When the information on the card is wrong, no matter what kind of format are written, there is no sense to discuss. In addition, there is not clear proof showing one format is better than the other one in terms of improving team performance[5]. Gojko also mentions to try different formats of a user story. For example, replacing words with a picture, using questions that spark people to discuss and to find solutions[5]. There are possible benefits of trying different story formats: some hidden creativity in the team might be woken up and a different perspective during the discussion could be discovered.

Thirdly, when write the stories, the short summary on the card should focus on user things rather than implementation and solutions. Clear and accurate descriptions of user roles can help the delivery team to identify needs and remove unnecessary complexity. It helps team to understand who will use the product they are building. As a result, it is important to identify the target user who will use the target system and investigate how work should be divided between them. User personae is recommended. A generic user role("As a user...") should be avoided. In addition, roles such as "As a tester" should also also be prevented.

B. Stories Discussing

User stories are reminders for discussion. There are following tips on user story discussion.

In terms of tools for the user story discussion, it is recommended to use whiteboard and flip-chart rather then using digital media. Discussions around the whiteboard are faster and more productive than with technical tools. Because using whiteboard does not need to worry about the text alignment issues and formatting as well as distractions. Big rooms with big boardroom tables are not recommended for discussion. This is because big tables with bigger room space prevents people from moving around and limits their physical interaction.

Talking about techniques on discussion, diverge and merge methods are encouraged to use in a large team. To make discussion efficiently, size of the group discussion should not be too big. It is suggested to divide the big team into smaller groups and let groups discuss their understand of a story first and then gather all groups together to compare results.

There are some benefits from splitting the teams. Splitting a large team into smaller groups minimizes a fake consensus. People with a similar background tend to work towards a consensus even the direction wrong[18]. It is faster to find multiple ideas for a story in a smaller size team as each member has time to contribute their ideas. In addition, everyone needs to compare their understanding with the others in the team. As a result, spotting sources of confusion is much easier. It also reduces risks on proposing unnecessary stories features. When comparing discussion results from different groups in the merging phase, it is easy to identify common sources of

misunderstanding. Comparing different solutions with different perspectives from the subgroups also enables innovations.

Experiencing of several times of splitting and grouping, all team members could have better understand of the content and scope of a story. Dividing and merging also makes the team work in parallel and explore alternative solutions in a faster way. When applying this method to story discussion, the team could realize and discover the correct solution faster and with relatively higher probability. Diverge and merge methods also help the team to avoid the heavily reliance on single source of all knowledge. One person who dominates the discussion in a single group is easy to happen.

In the diverging process, subgroups should use concrete examples to capture their understanding towards story. Because concrete examples are easy to compare and also easy to identify misunderstanding among different groups. When comparing the discussion results from different groups, the team should focus on below three differences:

- differences in the format or structure of information
- differences in outcomes of similar examples
- things scratched out and question marks...

The first one reflects mental models and the team needs to align different fromats and structures. The second one could show potential misunderstanding and they need to be discussed into further. The third one points out things that are might not clear to the other groups and needed to be discussed within the whole team.

A facilitator should be selected to keep track of the time and lead the discussion during the mering process. The facilitator should have the ability to compare results from different groups. That person should be obviously much more experienced and active in the business domain than the others so he or she also could easily spot misunderstanding and keep groups in the right direction.

Except tools and techniques for the discussion, discussion roles should also be taken into consideration when there is no need to gather all team members to look at a story. In such case, a smaller discussion with all relevant roles that representing each development, testing and analysis or other related field is necessary. This kind of smaller discussions are faster and require less effort on organizing people. Having all roles included ensures that all perspectives are covered and therefore ensures the quality and efficiency. It also enables other team members to contribute for the implementation, which makes the delivery more faster.

C. Stories planning

A good plan of user stories allows the team release product in time to obtain market opportunities with less competitors as well as saving cost. Releasing a product in a scheduled time plays an essential role in business success. This is because time, advertisement and business strategies are tightly associated with the release.

It is recommend to put a 'best before' date on stories. This is to avoid the team to leave stories to be implemented in the fire-fighting mode. When a team is reacting to an emergency, other work suffered and the quality of product might be degraded. To avoid emergencies, one method is to check if there is an expiry data when new user stories are proposed. The other approach is to write the best before data on stories and make them visually obvious. A clear expiry date allows teams to handle time-constrained stories before they become urgent. This not only reduces emergencies but also risks on developing lower quality product. It also help the team to work more effectively with normal pace. To make it work, it is important to ask about the potential deadlines and clearly specify the dates. Marking the item visually also helps the team to identify them quickly. Separating out fixed-date items by using different shaped or colourful cards is also recommended.

Similarly, except putting dates on the stories, it is also important to set explicit deadlines for addressing major risks among stories. Proper information on scheduling decisions also mitigates risks by avoiding hard and risky task from being post-phoned and replaced with small short-term wins. Alistair Cockburn[5] suggested that deliver team should focus on addressing risks rather then building value at the earlier stage, when risks are down, the team can focus on developing business value. Based on stages of growth model, the deliver team can decide when to deliver values and when to address risks as well as which tasks to address when product moves through different stages.

D. Stories prioritization

A collection of ordered stories for a software product is called product backlog. Prioritizing stories during the development process aligns decisions to business strategy and also associates user story implementation with business value delivery. This section will discuss methods on prioritize user stories.

Story card hell problem is a situation that a team has a bunch of user stories to be implemented but team does not have a big picture of what is going to do next. Using hierarchical backlog could keep things in a hierarchy which enables monitoring, discussing and reporting on big picture items. Arranging the backlog into tiers offers a big-picture view of the project and also reduces overall number of items. By using hierarchical backlog, the delivery team can spend less time on managing items and spend more time on building valuable product. In addition, the hierarchical backlog also allows the team to react quickly to market opportunities based on changes. For example, prioritizes could be changed at any level and a whole hierarchy of stories could be discarded when they are inapplicable. The visual connections among different levels also helps to show omitted features or unnecessary stories.

To implement the hierarchical backlog, a visual board with several horizontal swim lanes could be used to stand for different layers. It is also good to use a physical planning board with different card sizes and colors for different levels. User story map and impact are also good alternatives to help the team to present a hierarchical and multiple-level backlog.

Impact map is suggested to manage a group of stories. Figure 1 shows a structure of the impact map. It helps with prioritisation and eliminating unnecessary scope, which in turn

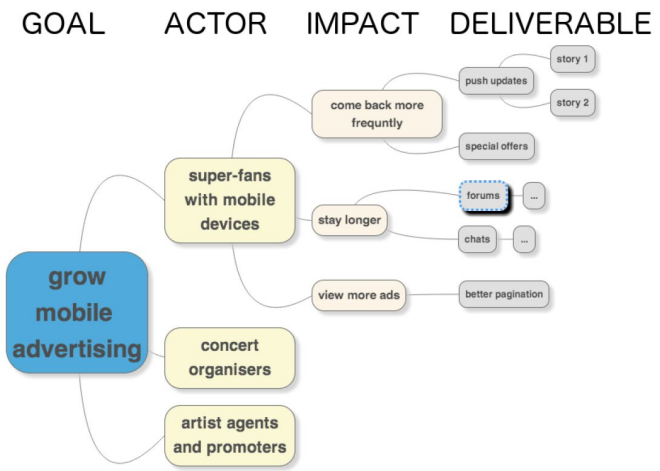

Fig. 1. Impact map structure[5]

speeds up delivery. An impact map is a strategic planning technique that helps teams align their activities with overall business objectives and make better road-map decisions[16]. Organizing stories based on impact map helps the team to make decisions and discuss prioritisation. With a single central node on the map, the stakeholder could choose one big business goal to delivery first. This not only helps with prioritization and removing unnecessary scope but also speeds up delivery. As information held in the card format is visually presented in the impact maps, user stories that do not fit the current release cycle could be easily spot and discarded. User story maps connect software deliver-ables (I want) to customer journeys and business work-flows. As a result, it shows how individual stories contribute to the bigger picture and offer a great visual representation of release plans. Story maps helps teams to have a better picture on what they are going to build and why they build that way. As visual tool, story maps help to spot missing or unnecessary user stories and create new product ideas. Story maps also helps with prioritizing and story splitting as well as release planning. There are several steps to make it work. Firstly, identify the backbone of the story map which is the horizontal axis. Then break down activities into high-level steps which should not imply a particular technology or a solution. After identify the backbone, identify options for stories and move items vertically to plan releases.

It is also suggested to prioritise user stories based on stages of growth. There are five stages of a growth model for success products[17]:

- Empathy: Figuring out how to solve a real problem in a way that people will pay for
- Stickiness: Building the right product to keep users around
- Virality: Growing user base organically and artificially
- Revenue: Establishing a sustainable, scalable business model with the right margins in a healthy ecosystem
- Scale: Growing the business...

At each stage, the delivery team should focus on addressing a particular set of risks. Making all the stakeholders to agree on the current stage is the most important thing to do when using this growth model. Then select objectives for current objects and identify quantifiable target metric that shows the team that the product is in the current stage or ready to enter another stage.

A standard method on prioritising a large group of work items is to use the MoSCow[5], which splits features into categories Must, Should, Could and Won't. However, majority of the work items will be put in the Must category. This is because the more the teams ask the stockholders to narrow down number of items in the prioritisation, the more the stakeholders are afraid to to leave those items out side of the backlog. To avoid this paradox, the Purpose Alignment Model is introduced[19]. Classifying items based on the Purpose alignment helps the team refocus efforts. The Purpose Alignment Model provides useful ideas on how to deal with category and avoid putting unnecessary items into Must Have category which causes complexity.

V. Summary and Conclusions

User stories are commonly used in agile software development and they stand for units of business value to be delivered. They emphasize on discussions to capture user needs and explore design solutions as well as spot misunderstandings. User stories must be verified by acceptance test before implementation. This requires the development team to build things that meet user needs and avoids creating unnecessary features in the upfront.

In this paper, overview of user stories is covered. Based on the background and history, Card, Conversation and Confirmation define the key elements of a story. The INVEST model shows some attributes of a good story. User stories alternatives such as requirements and use cases are also discussed to distinguish with user stories. User stories impacts on clarifying aim, reducing risks and costs, stimulating innovation are analyzed. Finally, tips on using user stories for business success are described based on referencing Gojko Adzic's book. In details, how to create, discuss, planning and prioritizing user stories are mentioned.

Talking about limitation of the paper, the paper lack of practical case study on user stories' impact towards a company. In the future, it would also be interesting to improve the paper on studying what kind of metrics are used to measure business value of user stories. Methods on evaluating user stories are also worth to research.

References

[1] Beck, Kent, Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 2nd Edition, 2004.

[2] Cohn, Mike, User Stories Applied: For Agile Software Development. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.2004.

[3] Jeffries.Ron. Essential XP: Card, Conversation, Confirmation. Available: http://xprogramming.com/articles/expcardconversationconfirmation/

[4] Wake, Bill,INVEST in Good Stories, and SMART Tasks. Retrieved October, 2014 Available: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

[5] A. Gojko and E. David, Fifty Quick Ideas to Improve your User Stories. Leanpub, Kindle Edition. 2014.

[6] J. Patton. User story mapping.O'Reilly Media. 2014.

[7] Sims,C. (2008). New User Story Format Emphasizes Business Value. [online] InfoQ. Available at: http://www.infoq.com/news/2008/06/new-user-story-format [Accessed 26 Nov. 2014].

[8] William Hudson.User stories don't help users: introducing persona stories. interactions 20, 6 (November 2013), p.50-53.

[9] I. Stamelos and P Sfetsos. (2007). Agile software development quality assurance. Hershey, PA: Information Science Reference, p.85-86.

[10] P.H.Deborah.(2006). Interview:Jim Johnson of the Standish Group. Available at: http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS [Accessed Oct. 2014].

[11] Lahanas. S.(2013). Aligning User Stories, Use Cases and Requirements - Dice News. Available at: http://news.dice.com/2013/01/15/how-to-align-user-stories-and-use-cases-for-your-requirements/

[12] Agilemanifesto.org. Principles behind the Agile Manifesto. [online] Available at: http://agilemanifesto.org/principles.html.

[13] L. Cao and B. Ramesh, "Agile Requirements Engineering Practices: An Empirical Study," IEEE SOFTWARE, Vol. 25, 01, pp. 60-67,JANUARY/FEBRUARY.

[14] Wells, D. (1999). User Stories. [online] Extremeprogramming.org. Available at: http://www.extremeprogramming.org/rules/userstories.html [Accessed 26 Nov. 2014].

[15] Agilemanifesto.org (2014). Principles behind the Agile Manifesto. [online] Available at: http://agilemanifesto.org/iso/en/principles.html [Accessed Nov. 2014].

[16] Adzic, G. (2012). Impact Mapping. [online] Impactmapping.org. Available at: http://impactmapping.org/about.php

[17] Croll, A. and Yoskovitz, B. (2013). Lean analytics. Sebastopol, CA: O'Reilly

[18] Surowiecki, James. (2004). The wisdom of crowds. New York: Doubleday.

[19] Nickolaisen, Niel. USING THE PURPOSE-BASED ALIGNMENT MODEL TOIMPROVE BUSINESS AND IT AGILITY.Available at: http://aamngt.com/files/purposebasedalignment.pdf

[20] M.Cohn.(2004). Advantages of User Stories for Requirements. Available at: http://www.mountaingoatsoftware.com/articles/advantages-of-user-stories-for-requirements

[21] M.Cohn.(2008). User Story Template Advantages. Available at: http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template.

[22] N.Dan(2003).Extreme Programming group.Groups.yahoo.com. Available at: https://groups.yahoo.com/neo/groups/extremeprogramming /conversations/topics/83897

[23] K. Conboyand C. OhEocha. The Role of the User Story Agile Practice in Innovation.

Creating shared understanding with Lego Serious Play

Juuso Hyvönen
Department of Computer Science
University of Helsinki
Email: juuso.hyvonen@cs.helsinki.fi

***Abstract*—Lego Serious play is a consultant service that utilizes the power of Lego bricks to create shared understanding and valuable goals for an organization. The method has been used by many large companies such as Nokia, Orange and Lego itself. Due to the metaphorical nature of the method, it also has a potential to be a cost effective way to create value for small and medium software organizations. This paper describes Lego Serious Play and its possibilities for creating shared understanding. The focus is on how the method supports stakeholder analysis and how the method could potentially help value creation and waste reduction in a lean software development framework.**

***Keywords*—Lean Software Development, Lego Serious Play, Stakeholder Analysis, Shared Understanding.**

I. Introduction

A typical agile software project consists of a product backlog, sprint backlog and sprints to develop the backlog items from the sprint backlog. The development model for the sprints is usually well documented and the best practices for each task during development have been defined. If they have not been defined, there exists a great deal of literature to support each phase of development, except to one very important part of the process, the black box called ”product backlog”. If the development items do not create value, all the great efforts to improve the development process are worthless. Finding the valuable tasks is usually left outside of the development process and the items are chosen by the managers and customer.

In my previous paper [1], I studied the use of Lean Canvas [2] and Impact Mapping [3] as tools for aligning product development to business goals. Lean Canvas was found to provide a great way of defining and clarifying a business model and Impact mapping was helpful in finding software development actions that make an impact to the user, and therefore help in reaching the business goals. One question was not answered, how are those business goals found and selected? One way could be increasing shared understanding. Schulz and Geithner [4] argue that to be able to collaborate effectively, individuals should have a collective shared understanding of the domain area they are working on.

This paper approaches the problem of finding meaningful business goals by offering a technique to improve shared understanding. A special emphasis is given for the viewpoint of stakeholders, the groups that share any interest to the project, product, or business in question. A technique for creating understanding is given in a form of Lego Serious Play (LSP) [5], a way of using Lego bricks to create real value for an organization by utilizing the knowledge of all the employees working for the the company. Focus of this paper is on how a small to medium lean software organization, such as a start-up, an intrapreneurial project or small business area of a company could potentially benefit from the use of Lego Serious Play. Bigger and more high-profile organizations might have to concider if the method discussed is thorough enough.

The outline of this paper is the following: Chapter two introduces Lego Serious Play, chapter three explains stakeholder analysis and introduces several techniques of analyzing stakeholders. Chapter four discusses if stakeholder analysis with Lego Serious Play is reasonable and gives an example of a real LSP workshop. Chapter five introduces lean software development methodology and connects LSP to lean principles. Chapter six covers the results and chapter seven concludes the paper.

II. Lego Serious Play

Lego Serious Play was developed at the The Lego Group in the late 1990s to help the middle and top managers of the company in strategic planning. The method was first used only internally at Lego but it was later developed into a full commercial consultant product. A deep description of how the method was born can be found in [6]. In 2010, part of the LSP method was made open-source. This paper uses the open-source part of the method as a starting point for study.

Lego Serious Play consist of three different kinds of resources [5]:

1) The Lego Serious Play basic principles and philosophy, upon which everything else is built
2) The Lego Serious Play materials - sets of specially selected Lego bricks and pieces
3) Lego Serious Play ”applications”- detailed roadmaps of different workshops which make use of the principles and philosophy and the materials.

The first two of these are the parts that were released to public in 2010.

LSP bases on an assumption that ”the answers are already in the room” [5]. It means that external experts are not necessarily needed and that the team participating in to a workshop is probably already capable of solving the problems in hand. The participants are asked to ”think with their hands”, meaning that they should build the ideas they possess. This helps in clarifying the concepts and demonstrating them to other participants and this way sharing understanding more

effectively than they could by just telling a story or drawing diagrams.

Lego bricks make it easy for the participants to build three dimensional structures to metaphorically resemble ideas in their minds. People, at least in western countries, are also typically already familiar with the bricks, and even if they were not, building Lego bricks is fast to learn because no high level of skill is required to be able to build structures out of the bricks.

The benefit of building things, according to Lego and [4], is that making something with hands makes the brain work a different way which can open new perspectives. An emphasis is also on the idea that leaders do not have all the answers and they need to hear the whole team and allow each member to contribute and speak out. There is a presumption that people naturally want to contribute and to be a part of something bigger and also take ownership. LSP gives everyone an opportunity to express their views of the problem and this way build a shared understanding of the problem. People might have very diversified views concerning the matter, and, especially, how to solve these issues. Often team members are not heard and the managers make decisions on their own, thus making the team work suboptimally.

There are a many different problems Lego Serious Play can be utilized to solve. Originally it was used to improve strategic work at the Lego Group [6]. Other documented use cases include using to elucidate project plan and different work roles in a long-term research project [4] and facilitating and pursuing on organizational interventions [7]. LSP has also been used in product development by capturing user stories and envisioning possibilities for service improvements [8].

Lego Serious Play is a consultant product which has the benefit that he facilitator is usually someone who does not permanently work in the company. LSP workshop always has an external facilitator. They takes care of all the arrangements required for the workshop and they also designs the workshop with a help of the customer company representative, for example, a product owner or project manager.

Lego Serious Play is practiced in a form of workshops. These workshops can be anything between three hours to multiple days in duration. The workshop consists of three phases that are repeated for the whole duration of the workshop. Phase 1 is called The Challenge. During that phase the facilitator gives a challenge for the participants. Phase 2 is Building. During that phase the participants build a Lego model that, they think, represents an answer to the challenge. The answer can be a concrete answer or it can be a metaphor that represents the answer. In phase 3, Sharing, everyone shares the meaning of their model. Explaining the answer can include telling a story by moving Lego parts around the structure (playing seriously) to visualize their thoughts. After part 3, the facilitator poses the next challenge and the whole process repeats. The workshop starts with simple challenges like ”Build a tower” to make everyone familiar with the bricks and proceeds towards more challenging tasks. In the end, cooperative challenges can be presented.

III. Stakeholder analysis

Stakeholders are persons, groups and organizations that have to be taken into account when business decisions are made [9]. Other common definitions according to Bryson [9] include:

- ”Any person group or organization that can place a claim on the organization’s attention, resources, or output, or is affected by that output” [10]
- ”People or small groups with the power to respond to, negotiate with, and change the strategic future of the organization” [11]
- ”Those individuals or groups who depend on the organization to fulfill their own goals and on whom, in turn, the organization depends” [12]

The definitions above share a similar meaning that stakeholders are groups that count when decisions about any business activities are made. Software business being a business like any other, the stakeholders are as important to software business as they are to any business. Stakeholders are something that can create or reduce the value of the software system. This makes stakeholder analysis a necessary tool for value creation.

Stakeholder analysis can be expensive and time consuming process because of the high amount of people and time it can require. Performing excessive stakeholder analysis can be seen as waste of time and money. LSP could potentially be a method that supports understanding stakeholders while getting other kinds of shared understanding simultaneously. Finding the proper level of stakeholder analysis and the proper techniques to do it are essential.

Bryson hypothesizes in [9] that ”strategic management processes that employ a reasonable number of competently done stakeholder analyses are more likely to be successful – that is, meet mandates, fulfill missions and create public value – than those that do not”. He also states that what constitutes to ”reasonable” amount of stakeholder analysis is not clear, yet. The idea is similar to a lean principal of reducing waste which will be introduced later in this paper.

Making excessive stakeholder analysis is waste and the biggest benefit can be achieved when a right, not too much, nor too little, amount of planning is executed, as bryson just stated . Sometimes it might be enough to make project team aware that there are multiple stakeholders and the other times a deep and systematical stakeholder analysis could be beneficial. The systematic approach is especially important for high value public service projects as they often directly affect people’s lives and get a lot of public coverage.

Bryson analyses mapping techniques from the point of view of a public organization but this paper focuses on the ability to provide value for lean software development. Public organizations usually map the effects of their actions far more systematically than small and medium private companies. To create enough context for stakeholder analysis, this systematical and deep, public organization -level of analysis is introduced first. All the techniques introduced require no equipment outside typical office facilitation materials, such as,

flip charts and post-it notes. The biggest costs for executing these methods comes from the personnel fees.

It is important to understand that systematical analysis of stakeholders using formal methods can give better results compared to less vague methods such as Lego Serious Play. But because the cost of implementing a multi stage analysis of stakeholders and their issues is quite high, the value gained from the analyses might not cover the expenses.

Bryson groups stakeholder analysis techniques into four different groups [9]:

1) Organizing participation
2) Creating ideas for strategic interventions
3) Building a winning coalition around proposal development, review and adoption
4) Implementing, monitoring and evaluating strategic interventions

Bryson uses a directed graph to visualize how different stakeholder analysis methods support each other and how they support strategic management. Original visualization used the term ”public value” but in our visualization a more abstract definition of just plain ”value” is used because value can be interpreted in numerous different ways [13][14].

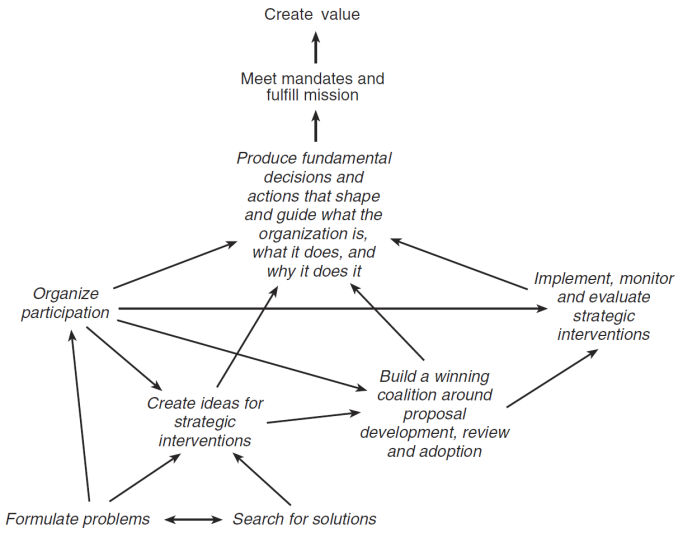

Fig. 1. How stakeholder analysis methods support value creation [9]

Next, stakeholder analysis method groups mentioned above and some of the methods are given a brief overview. After that, they are compared to Lego Serious Play.

1) Organizing participation focuses on finding out who should be involved in the different stages of the analysis process. The first group also includes techniques to find the actual stakeholders. The rule of the thumb is that a people should be invited if they have information that can not be gained without their participation.

The basic stakeholder analysis technique is brainstorming, with analysis added to complement the findings. A list of stakeholders is created and a separate flip chart sheet for each stakeholder is prepared. For each stakeholder, their expectations for the organization (or a software system) are listed and the stakeholders judgement of the organization is decided: good (green), fair (yellow), poor (red). What can be done to satisfy the stakeholder and the possible long-term issues that the stakeholders could face with the business are identified. Additional interesting information such as specifying how each stakeholder group influences the organization and what the organization needs from each stakeholder is listed. The stakeholders can also be ranked according to their importance for the success of the business.

Power versus interest grid supplements the basic stakeholder analysis technique. It is used to visualize and determine what stakeholder groups' interests should be taken into account, what kinds of interest coalitions might be present in the stakeholder groups and what kinds of actions the organization should prepare for to satisfy the stakeholder needs.

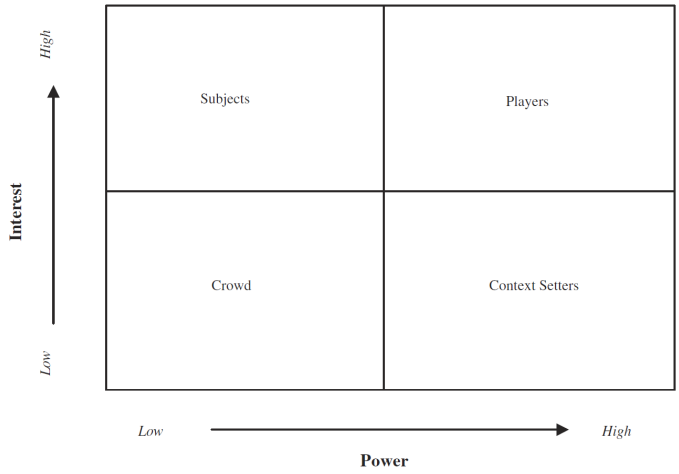

Fig. 2. Power versus interest grid [9]

Executing the technique is not complicated as it only requires a grid drawn to any surface, for example on to a whiteboard and post-it notes to resemble the stakeholders. The vertical axis resembles interest and the horizontal axis resembles power of the stakeholder. Each stakeholder is then placed to an appropriate place in the grid and the labels are moved until everyone is satisfied with the placements. After the process is ready, a discussion about the implications of the results is held.

Stakeholder influence diagram visualizes how stakeholder groups influence one another. This analysis can be done by placing the stakeholder groups as post-it notes on a whiteboard and drawing arrows that indicate the influence and it's direction between the entities. The arrows can be named to clarify the type of influence.

2) Creating ideas for strategic interventions focuses on understanding the stakeholders and formulating potential problems and searching for solutions. Bryson also mentions the focus on political feasibility which can be an important focus point when the analyzer is a public organization or if the software system could potentially have issues with legal regulations or ethical rules.

Stakeholder-issue interrelationship diagram builds on top of Power versus interest grid, Stakeholder influence diagrams

and Basic stakeholder analysis technique. Stakeholder influence diagram helps to understand which stakeholders have interest in which issues and which groups share the same interests. The result can help, for example, in finding the potential areas for cooperation and conflicts.

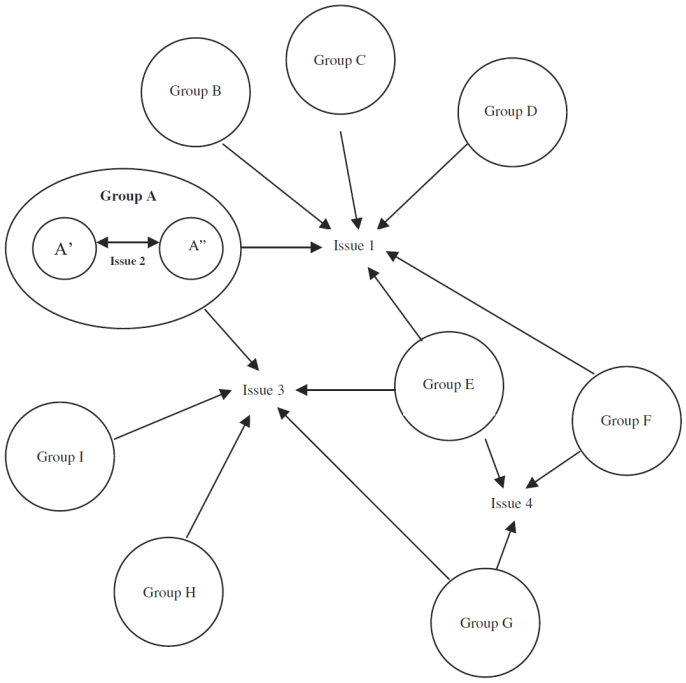

Fig. 3. Stakeholder-issue interrelationship diagram [9]

The method starts with writing down the issues found earlier on post-it notes. Then the issues are placed on whiteboard or similar. After that, stakeholders are placed around the issues. One stakeholder can have connection to many issues. After the stakeholders and issues are set properly, arrows are drawn to indicate how issues and stakeholders are connected. After the map is complete, a discussion about the results should be held.

3) Building a winning coalition around proposal development, review and adoption focuses on additional analysis of the issues found at the previous stages of analysis.

The most straightforward way Bryson proposes for proposal development and review is *stakeholder roleplay*: Each participant reviews the previously created analyses. After that, for each member, a role is given and each participant reviews the issues for their role. The participant's task is to answer the questions "How would I react to this option?" and "What would be done that would increase my support or decrease my opposition?". The answers should be written to sheets and the exercise should be done multiple times with randomly drawn roles to increase the quality and diversity of the answers.

4) Implementing, monitoring and evaluating strategic interventions focuses on making sure that the decisions made earlier are kept and the solutions found implemented.

Policy Implementation Strategy Development Grid is used to create an actionable table of actions that should be done to satisfy stakeholders. All the material from the stakeholder analysis should be used to populate the grid.

Stakeholders	Stake or Interest	Resources	Action Channels Open to Stakeholder	Probablity of Participation and Manner of Doing So	Influence – as a Product of Resources and Participation	Implications for Implementation Strategy	Action Plan Elements
Supportive Stakeholders							
Opposing Stakeholders							

Fig. 4. Policy Implementation Strategy Development Grid [9]

Bryson also introduces several other analysis techniques but they are left out of this discussion because of their focus on issues that are more vital to bigger companies and public organizations.

IV. Stakeholder analysis with Lego Serious Play

Stakeholder analysis with Lego Serious Play utilizes the potential of all the participants of the workshop for building a shared understanding. Schulz and Geithner studied LSP as a tool for building shared understanding for a researcher team [4]. They found out that the combination of a brick model and story telling added value compared to traditional forms of group meetings. This was partly due that that all the parts of the brick model are named and explained and that the participants can ask questions about the model to avoid misunderstandings. Schulz and Geithner also found out that building a physical model helped in reifying and reflecting the builder's own understanding. They also argue that even though it could be said that the Lego model is too static, the model is only an anchor for a metaphor and it can be used to connect a story to the model to provide meaning.

Schulz and Geithner also claim that the first part of the workshop, the individual challenge solving, is about expression of personal understandings, the individual awareness of them and making their understanding explicit to the others. The cooperative part of the workshop, building a shared model, can be seen as building a shared understanding. Building a shared model forces the participants to modify their views to make them fit together. The final solution will then represent all the different views on the same issue and that way it provides a shared understanding which can be utilized for innovation.

It is possible to create many connections from Lego Serious Play to different stakeholder analysis methods. The focus of

the LSP workshop can be set to finding stakeholders but as noted earlier, LSP is not specifically designed to systematically map stakeholders. The potential of stakeholder analysis lies in utilizing the knowledge of all the participants and creating revealing metaphors of the business area. The focus of the workshop can also be wider subject such as building an unified vision of a product. Even when stakeholder are not the main focus point, many aspects of them get analysed.

To find simple and concrete links to the previously introduced ways of analysing stakeholders, following connections can be seen:

Organizing participation can happen during any time of the workshop. For example participants could refer to different stakeholders during their sharing turn. In the final unified model, different stakeholder groups and their relation to the business get presented. What seems to happen is that the stakeholder's power gets represented with easily understood metaphors. For example, customers could be sheeps and CEO could be a skeleton with a whip standing on a pile of coins.

LSP powerfully visualizes stakeholder-issue interrelationships and therefore is helpful in *creating ideas for strategic interventions*. Lego even sells a separate LSP Connection Kit to aid in networking different parts of the Lego structures.

When participants share their metaphors, they often role-play some important role related to the business and therefore participate in *building a winning coalition around proposal development, review and adoption*. LSP does not directly give tools to help *implementing, monitoring and evaluating strategic interventions* but during a workshop, ideas how to implement, monitor and evaluate can come up. For example new (or existing) ways to communicate to customers can be present in the final cooperative model.

To give an insight, what a real LSP workshop is like, rest of this chapter describes a real LSP workshop that took place in a medium-sized Finnish software company.

The session was facilitated by two researchers who specialize in service design. They had self learned the LSP method and were not LSP certified professionals. The participants of the workshop included the product owner, a team leader, three developers, one salesperson and one product support person. The duration of the workshop was 6 hours. There was a lunch break and two short coffee breaks during the session. The goal of the workshop was "To create a shared vision of the Product".

The workshop consisted of the following challenges:

1) Build a tower
2) Build your ideal neighbour
3) Build your typical Monday
4) Build a representation of a user of the Product.
5) Build an important challenge of your user
6) Build a solution to that challenge.
7) Build your vision of the future 'Product' in an ideal world.
8) Now, as a team... build an answer to the previous challenge

All the models build during the workshop were photographed with a description note next to the model. After the workshop, the facilitators produced a booklet which summarized the workshop from their point of view. The booklet was distributed to all the participants of the workshop.

Fig. 5. A typical Monday

Participants were mostly excited about the new method. Only one participant of LSP stated after the workshop that he did not like the idea of using Lego at all. All the others gave very positive feedback. The Lego bricks also gathered a lot of interest inside the company. An another workshop for another product development team was ordered immediately after the first workshop.

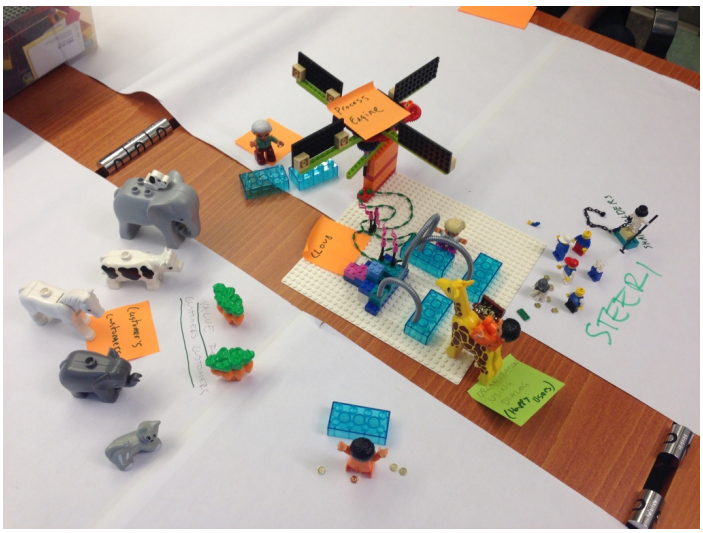

Fig. 6. The shared vision of the future

Multiple problems in the product and in the development process were found. The emphasis of the workshop was not on finding the stakeholders but different stakeholder groups became very clear during the last challenge. Even one new stakeholder group was found. Also some thoughts of how this group could be taken into account during product development were expressed. The findings from the workshop were utilized when deciding where the focus of the product development should be in future. A prototype project was launched to test

if the new direction is feasible.

V. Lego Serious Play and Lean software development

Lean software development [15] is a software development method adapted by Mary and Tom Poppendieck from Toyota Automobile Production System [16]. It does not compete with agile development methods but is rather a complementary philosophy to support producing value and reducing waste. Lean software development consists of seven principles which one should always be considered when making decisions. The principles according to Poppendiecks are:

1) Eliminate Waste: All actions that do not create value are considered waste. This could be seen as the main Lean Principle as the rest of the principles could be derived from eliminating waste. Eliminating waste means, for example, avoiding extra software features, avoiding partially done work and avoiding waiting.

One major problem in finding the waste is that different interest groups inside the company might have completely different value expectations for the same product [13]. Software developers merely want to make a product that works and is of high technical quality. Product and project managers often have more high level vision, and sales department might expect value to be something that can be clearly communicated to potential customers. High level of shared understanding provided by Lego Serious Play could help in eliminating waste.

2) Amplify Learning: Amlifying learning means that the team tries to constantly improve their understanding of their area of business and their product. This can mean building feedback loops as short as possible. which includes fast testing cycles and short iterations, as well as fast customer feedback. While those methods to amplify learning are merely tools to steer the development, a LSP workshop is method to improve and share the domain knowledge of the whole product team.

3) Decide as Late as Possible: If decisions are made too early, there might not be enough data to enable wise decision making and waste could be created. If decisions are made too late, waiting happens and waste is generated again. For software developers, this can be described as Just-In-Time decisions comparable to Just-In-Time compilation. A shared understanding and a good understanding of stakeholders create by LSP could mean better decisions but it's difficult to argue that they would make it possible to delay decisions even more.

4) Deliver as Fast as Possible: When decisions are made as late as possible, software delivery needs to be fast. Rapid delivery simply makes it possible to make important decisions later than with less rapid delivery. Rapid delivery is achieved by utilizing the time spent at work as effectively as possible. It is debatable if an understanding of the business area has a straight correlation to reduced time spent coding a feature but if the point of view of speed of delivery is changed from features to goals, improvement might happen. Understanding the domain area better, should make it easier to choose the right features to reach the goal and this way reduce waste as fewer iterations are needed to reach a goal. Likewise, Understanding the business area could potentially correlate with better maintained code base which could reduce the actual time spent implementing the features. The principle of building integrity in will explain this more.

5) Empower the Team: Traditionally managers tell employees what to do and how to do their work. Lean software development sees this relation in a completely different way. Workers are seen as experts in their own field and managers job is more about making it possible for the team to use their expertise the best way possible. LSP takes this idea a step further and call is "The Answer is in the system". Besides being experts on their field, LSP acknowledges that in a highly educated organization full of specialists, the people understand a lot of things outside their own field of expertise. LSP is aiming to harness that expertise to be used in decision making.

6) Build Integrity in: Customer's perceived integrity is about the whole experience with the product. The customer should feel that all features are relevant and that they work as they are expected. Conceptual integrity is all about understanding the business domain and creating components that work well together. The key for integrity is understanding the whole system. LSP seems to support this principle as it effectively helps in creating a shared understanding. When the business area is understood well, the integrity of the product increases and the customer should also be able to feel it. Integrity can also be effectively build into code base which can reduce development times and that way reduce waste.

7) See the Whole: "A system is not just the sum of its parts - it is the product of their interactions." [15] Optimizing one part does not mean that it will improve the system. The whole system should be understood, and only in that way the interaction between all the components of the system can be optimized to create the best system possible. Like with the previous principles, LSP is a way to create shared understanding and to understand the domain area.

As the analysis shows, Lego Serious Play fits well into lean software development framework. The method is also quite lightweight and fun to use which makes makes it very approachable. LSP is not best choice when systematical analysis must be made but it works great when an overview of some domain should be created. Example use case could be project kick-offs to create a mutual understanding. It could also be used to improve the unified vision of a product for example once or twice a year.

VI. Results

Lego model and storytelling can potentially create more shared value compared to many other forms of planning. Lego Serious play bases on the presumption that a participant builds the idea he or she has and this way makes his or her idea concrete. This gives them opportunity to explain themselves with more than just words. The concept and environment of the workshop is very playful which also makes participants very open to discussion. Even opinions that participants would not dare to say elsewhere can be spoken out. This helps in sharing the understanding of the products, the issues, the stakeholders and the business in overall.

The external facilitator seems to make the atmosphere of the workshop better as the leader is now not someone every participant should try please. Compared to for example drawing or clay, Lego bricks have the advantage of the participant

not needing to be skillful in those more traditional handcraft methods. Also compared to clay, Lego bricks are quite easy to clean after the workshop ends.

Limitations of the method are that there is a risk of moving away from the original task to focus too much on the Lego model, not the challenge in hand. Participants would easily start focusing on finding certain special Lego bricks instead of focusing on the metaphor. Having a documented way to document describe the findings of a workshop would be helpful. Now it if left for managers to choose what findings are important. The lack of documentation on documenting the process results could potentially lead to a situation where no concrete actions or results can be derived from the workshop. The workshop at the case company verified these shortcomings.

VII. Conclusions

This study introduced Lego Serious Play which seemed to be potentially an effective way to create shared understanding. LSP, as a tool to create shared understanding, seems to nicely fit into a lean enterprise to support value creation. The concrete measurable value of LSP was not studied. LSP is useful for analyzing the business area and finding unmapped parts of the domain. LSP has a direct connection to stakeholder mapping but is not well fit for systematical analysis. Instead, it is better used to create overviews and visions for the product team. Lego Serious Play can help in creating more sound and valuable software systems.

References

[1] J. Hyvönen, "Lean canvas and impact mapping as tools for linking product development to business goals – a case study," in *Real-Time Value Delivery in Software Engineering*, J. Münch, Ed., 2014, pp. 20–27.

[2] A. Maurya. (2012) Why Lean Canvas vs Business Model Canvas? [Online]. Available: http://practicetrumpstheory.com/2012/02/why-lean-canvas/

[3] G. Adzic and M. Bisset, *Impact Mapping: Making a Big Impact with Software Products and Projects*. Provoking Thoughts, 2012.

[4] K.-P. Schulz and S. Geithner, "The development of shared understandings and innovation through metaphorical methods such as lego serious play^{TM}," 2011.

[5] LEGO, *Introduction to LEGO Serious Play*, 2010.

[6] J. Roos, B. Victor, and M. Statler, "Playing seriously with strategy," *Long Range Planning*, vol. 37, no. 6, pp. 549–568, 2004.

[7] M. Statler and D. Oliver, "Facilitating serious play," *The Oxford Handbook on Organizational Decision-Making (Oxford University Press, Oxford)*, pp. 475–494, 2008.

[8] D. Swann, "Nhs at home: Using lego serious play to capture service narratives and envision future healthcare products." *INCLUDE 2011 Proceedings*, 2011.

[9] J. M. Bryson, "What to do when stakeholders matter: stakeholder identification and analysis techniques," *Public management review*, vol. 6, no. 1, pp. 21–53, 2004.

[10] ——, *Strategic Planning for Public and Nonprofit Organizations: A Guide to Strengthening and Sustaining Organizational Achievement*, ser. Jossey-Bass nonprofit sector series. Jossey-Bass Publishers, 1995.

[11] C. Eden and F. Ackermann, *Making Strategy: The Journey of Strategic Management*. SAGE Publications, 1998.

[12] G. Johnson and K. Scholes, *Exploring Corporate Strategy: Text and Cases*. Financial Times Prentice Hall, 2002.

[13] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher, Eds., *Value-Based Software Engineering*. Berlin: Springer, 2006.

[14] J. Heidenberg, M. Weijola, K. Mikkonen, and I. Porres, "A model for business value in large-scale agile and lean software development," in *Systems, Software and Services Process Improvement*, ser. Communications in Computer and Information Science, D. Winkler, R. O'Connor, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2012, vol. 301, pp. 49–60.

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

[16] T. Ohno, *Toyota Production System: Beyond Large-Scale Production*. Cambridge, MA: Productivity, 1988.

Review: DevOps, value-driven principles, methodologies and tools

Tuukka Peuraniemi
Department of Computer Science,
University of Helsinki
Helsinki, Finland
tuukka.peuraniemi@gmail.com

***Abstract*—This review aims to clarify definition of the term DevOps and what are its value-driven principles, methodologies and tools. DevOps combines operations and development while extending agile methodologies and principles outside development [3]. It adds a more value-driven and customer centric approach on agile software development while they share many principles and methodologies. Four main principles adopted from agile are: respect for one another, commitment to shared goals, collective ownership and shared values [1]. DevOps emphasizes heavily on continuous delivery, amplified feedback, more-stable infrastructure and monitoring and metrics. Continuous delivery consists of building, testing, deploying and version management. This makes deployment less risky and faster. More-stable and reproducible infrastructure makes continuous delivery possible and makes use of DevOps tools of virtualization and cloud computation. DevOps has brought forth new kinds of tools e.g. New Relic for monitoring and real-time data analytics, Vagrant lightweight and portable virtual machines, Puppet and Chef configuration management for development environments and finally Docker for application containerization without the need of creating new VM for different platforms. DevOps approach is value-driven as it enables faster feature delivery and shortens the lead time and time to market. Also more stable operating environments have direct impact on service availability. Enabling developers to spend more time developing new features and less time on fixing and resolving problems. More time can be spent on adding value to the product. These previously mentioned factors enable bigger market shares and competitive advantage. Both large and small companies benefit from DevOps approach.**

Keywords—DevOps, methodologies, principles, tools, value-driven, customer satisfaction

I. INTRODUCTION

DevOps is an ambiguous term used in many different contexts and with different definitions. Many sources define DevOps as a combination of development and system operations. DevOps can be viewed as a way of managing software's lifecycle. It also extends agile methodologies from development to system operations. Combination of both fields leads to face the conflicts of operations "*fear of change*" and developments "*need for change*" [1]. DevOps shares many principles with agile methodologies and is often referred as "*agile on steroids"* but it also includes some Lean principles. Before processes can be automatic it should be compact and review according to Lean principles. Figure 1 depicts on DevOps relation to software development and operations and how it indirectly has impact on business performance.

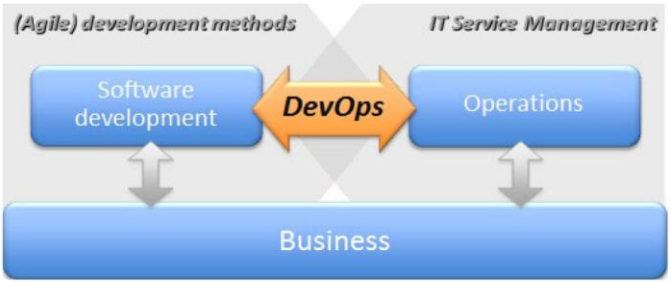

Figure 1. DevOps relation to operations and software development and how it indirectly relates to business [1].

Usually DevOps is involved in web services and cloud based services including infrastructures, platforms and software as a service (IaaS, PaaS and SaaS). DevOps has four different aspects: culture, automation, measurement and sharing [1]. Culture meaning people over processes and tools, automation for quick feedback, measurement for quality control and finally sharing tools and processes with others. According to DevOps lifecycle of the software should be as automatic as possible from test to deployment. Hence replacing agile methodologies' continuous integration with continuous delivery and aiming for continuous development. Continuous delivery consists of automatic build, deploy, test and release. Difference between continuous delivery and continuous deployment is that in the first before deploying the product has to pass manually done acceptance tests whereas the latter does automatic acceptance testing. Continuous deployment is typical for small companies and startups and continuous delivery is usually done in larger companies. DevOps has already brought forth many different lightweight tools with strong emphasis on cloud based tools for managing different states of software life cycle. E. g. Capistrano for easy and rapid deployment or New Relic as a SaaS for monitoring performance [11].

DevOps creates value from customer satisfaction by reacting quickly to customers requests shortening feature lead time and faster time to market. This also enables getting rapid feedback from the customer and helps with recognizing customers needs. Customer satisfaction with services can be related to three different elements: core service quality, relational service quality (delivery) and perceived quality [4]. Concerning web and cloud services DevOps has effect on all three areas linked with services customer satisfaction. Customization and fast delivery is essential for customer satisfaction especially in fast-growing and client-focused companies [5].

This article aims to clarify the definition of DevOps and briefly reviewing its principles, methodologies and tools from value-driven perspective. What are the characteristics of these areas which give value to the customer and how it delivers it. Also this article aims to specify on which of the three areas of services customer satisfaction its principles and methodologies apply.

In the second section DevOps definition is given with contrast to other definitions. Third section concerns DevOps principles and of which can be seen as value-driven principles. In the fourth section DevOps methodologies are briefly summarized. Fifth section concerns on different tools which are used by DevOps with strong emphasis on value-driven aspects which are analyzed further in discussion. In the sixth section a case study on applying DevOps principles in a large company such as IBM [2]. Seventh sections concerns discussion with comments and in the final section is for the conclusions.

II. DevOps - definition

DevOps extends agile principles and methodologies to system operations [1,3]. It tries to break down silos and barriers of separate IT teams. This is done by extending developers workflow from development to service delivery. From a system operations perspective workflow extends from operations to development. Depending on which roles workflow is extended one can have more weight on either end of the workflow. But the main goal is to have a clear picture of the whole life cycle of the software thus making it easier to define what is needed in the next steps (fig. 2) [15].

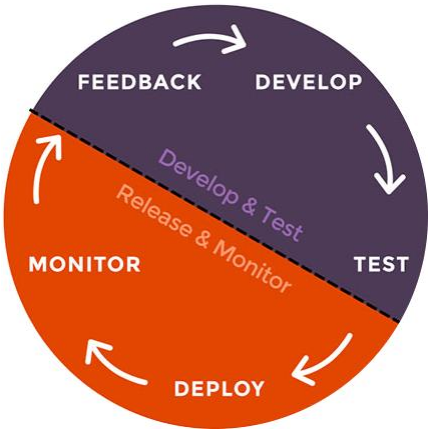

Figure 2. DevOps way of managing lifecycle and rapid releasing [15].

DevOps lifecycle extends developers and system administrators responsibility scope and view to include everything from development to delivery and monitoring [15]. Typical lifecycle is as follows. After new feature or bug fix has been developed it is tested automatically and if the tests pass and the build is successful it is automatically deployed. When it is deployed data is collected from the deployment and for the whole product giving DevOps lifecycle also a data-driven view. Useless features, bottlenecks and other problems can be identified from the product or the infrastructure and from this information new design decisions and other changes can be made. This together with customers feedback enables good adaptability and increases customers satisfaction thus adding more value to the product.

When mixing development and operations conflicts arise e.g. stability vs. features [3]. DevOps tries to minimize developments and operations incentives by unifying them. It also helps resolving certain conflicts by making it easier to prioritize problems and possibly solving it in another state of the softwares life cycle. E. g. connecting new components to application with deployment tools. DevOps helps with softwares architecture and design solutions by making it easier to identify possible problem scenarios in development and deployment thus avoiding risky deployments [1].

III. Principles

DevOps shares many of its principles with agile software development extending it from software to more general aspects of software delivery and customer satisfaction [3]. It delivers value to the customer with faster delivery and better modifiability. One of most common principle associated with DevOps is infrastructure as code. Hütterman has listed four general principles of collaboration which is the main focus of DevOps [1]:

1. Respect for one another
2. Commitment to shared goals
3. Collective ownership
4. Shared values

Respect for your team members and colleagues values individuals and interactions over processes or tools. The last three of the collaborational aspects are more value-driven than the first one. Commitment to shared goals makes helps with team dynamics and making sure that everybody does his part for the project with all states of software life cycles in mind. Collective ownership makes sure that everybody uses the most suitable solutions when developing software enhancing the perceived quality of the software. E.g. it enforces service stability which increases the services availability. Also the whole team performs quality assurance including both development and operations. Aspects of quality assurance are included in continuous integration and delivery e. g. build has to pass and no failing tests. When all of the team members are familiar with the software they can make customizations or bug fixes independently and deploying them. Shared values and commitment to it helps with prioritizing fast delivery, working system and customer satisfaction over contract negotiations or formal plans. This increases services modifiability which in turn delivers value to the customer. E.g. increased modifiability speeds up adapting to new business and technology challenges making it possible to gain advantage over competing services and businesses. Last three principles have effect on all three areas of services customer satisfaction. All of these principles aim to give developers a more holistic view of their doings [1,8].

While DevOps principles are similar to agile software development it mixes in Lean aspects and emphasizes on different areas and scope than agile [7]. DevOps has Lean aspects as it concentrates on delivering continuously more value for the customer and also by making deployment easier and testing automatic it frees developers time for developing features [14]. Main principle is communication and collaboration of development and operations. Trying to find balance on infrastructure and development results in better stability without the expense of development. Extending agile principles outside development towards operations makes managing value streams enabled by IT easier. Finally DevOps implements culture of continual experimentation and learning applies all of its principles and reviews what succeeded and what didn't. In the next iteration goal is to resolve previous iterations problems and improve in weaker areas. This kind of continuous improvement and amplified feedback enforces value-driven development.

A. Infrastructure as code

Infrastructure as code (IaC) refers to automated configuration management. This can be achieved using suitable programming language in making the configurations in behalf of the developer [1]. Managing configurations automatically makes the process more efficient, repeatable and faster. Coding the configuration also helps understanding the flow of the dependencies and at the same time "documenting" it. This also helps with making changes to the infrastructure easier. Typical tools for infrastructure as code are Puppet, Chef and Vagrant which are explained in the fifth section, tools.

This also creates new problems. There are many restrictions concerning Windows environments whereas Linux is much more flexible. Some tools are only available on linux platforms e. g. Docker. When designing infrastructure one has to take in account different tools available for the platform in question and many of the tools have a certain learning curve and their own syntax.

IaC makes configuration of environments faster thus delivering services faster for the customer. Stabile infrastructure makes deploying less risky lowering failure rates. Infrastructure that enables better feature flow lead time is shortened. If something does go wrong recovery is from it is faster. Fixed or rollback to previous stable version is easier with DevOps tools. This brings value as both perceived and core service quality. Infrastructure is in the core of the software making its development and deployment easier.

IV. METHODOLOGIES

Many of the methods and processes involved in DevOps are the same as in agile software development e.g. Scrum, Kanban [1,3]. DevOps methodologies add value to the customer in the same way as in agile development. Difference is in the nature of tasks in the Kanban board or in the backlog which differ from development to operations. Also the definition of done can be extended from tested and integrated to delivered. This emphasizes the nature of fast delivery and customization when all of the tasks can be seen directly in customer satisfaction. DevOps also enforces the methodology of small and frequent releasing (fig. 3) [1]. This makes feature delivery more stable not creating large gaps on functionality between versions. Customer can see where the service is heading and if he thinks that it is developed in the right direction.

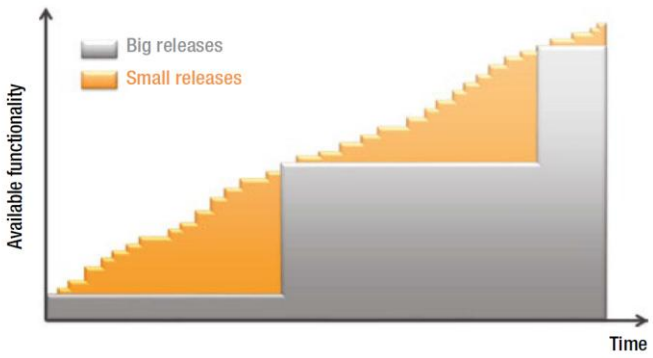

Figure 3. DevOps methodology of release often, release small compared to traditional release control [1].

DevOps methodologies though indirectly are value-driven (fig. 1). Faster development cycles adds new features faster and making the operating environments more stable adding perceived and relative quality to the product. Also when operating environments are more stable it saves time from maintaining and fixing the environment to concentrate on adding value to the product. These business side benefits are tightly bound with technological side. This requires technical side to implement rapid continuous software delivery and resolve problems faster.

DevOps has also sprung new methodologies e. g. visible ops, which is based on observational data from hundreds of IT organizations [6]. Visible ops aims to distinguish high-performing organizations and benchmarks its progress. Process is distinguished in four phases. First phase called "Stabilize the Patient" concentrates on problem management applying the changes to the software in a scheduled manor minimizing the number of outages. Second phase is "Catch & Release" and "Find Fragile Artifacts" as it emphasizes configuration and change management and introduces IaC tools and fragile artifacts are identified and handled accordingly. Phase three concentrates on release management processes as it aims to create repeatable builds. After this step it is easier to rebuild the software than trying to repair it. Fourth and final phase emphasizes on iterating the previous steps for continuous improvement and implements monitoring tools for measuring its success. Relying on the principle "You cannot manage what you can't measure".

Visible ops helps with continuous delivery and success rate of customizations as the continuous monitoring keeps record of the performance. Each step is designed to create value in improving old processes and creating new more suitable ones. Thus visible ops can be seen as a value-driven methodology. When contrasting the visible ops with services customer satisfaction it concentrates on relational service quality and from that viewpoint aims to improve the overall quality.

V. TOOLS

DevOps draws many of its advantages from virtualization and cloud technologies. Tools can be roughly divided into two groups: metrics, improved feature flow and delivery [1]. Measurement and metrics are important in recognizing bottlenecks, ensuring service and feature quality and it can also be a driving factor in feature development. Recognizing problems and areas that need development delivers value to the customer in better service quality and new features. One tool for service monitoring is New Relic [11]. Chef and Puppet are configuration management and infrastructure provisioning tools. Vagrant can be used to set up virtual machine (VM) environments for development and testing [10]. Docker automatizes deploying of applications inside software containers [9]. All these tools can be used together creating a measurable and easy-to-configure platform for services.

New Relic is a SaaS and it is also involved in software analytics [11]. It can be used in most of the databases and web service frameworks e.g. Django and Ruby on Rails and it is easy to install. Instrumentation to the service itself is relatively lightweight not throttling the performance of it. One can monitor code level performance of the service giving insight on possible bottlenecks ensuring future service quality. Many services have a large infrastructure of other services and servers and it can be difficult to get the whole picture of the infrastructure let alone monitoring it. New Relic can be used for the whole infrastructure and in making so one can find problems and new optimizations for it. Knowing services and databases response time increases service availability and perceived quality.

Chef and Puppet are two different approaches to manage configuration [12,13]. Configuration management makes controlling environments easier and deploying automatically possible. They differ both in syntax and approach where Puppet is model-driven concentrating on dependencies and Chef is procedural involving install scripting with Ruby based DSL (Domain Specific Language). Puppet uses its own JSON-like language. From operations perspective Puppet seems more familiar to use than Chef and vice versa for developers.

Docker is a lightweight open platform which acts as a container where processes run in isolation [9]. At the moment Docker supports only linux environments. Normally VMs are used as a platform for virtualized applications to run on. This however causes a large overhead just for running the application. One has to install all the dependencies, drivers and libraries for the whole operating system. Docker unifies the virtual platform with Docker Engine. It offers an high-level API (Application Programming Interface) and enables applications to be run without the operating system layer. This makes makes Docker more portable and efficient than traditional VMs.

Vagrant is a VM manager making its configuration easier [10]. Where Docker creates a unified platform for applications Vagrant creates a full VM. Versus traditional VM configuration Vagrant makes creating and configuration of VM reproducible enabling source control through one configuration file. Depending on the environment needs one can choose from Docker which is linux-only and more universal Vagrant. One can also combine the versatility of both platforms where Docker is run inside Vagrant configurated VM. Dockers dependencies and configurations can be made with Puppet or Chef.

VI. IBM CASE STUDY

In this case study we first review the situation in the beginning, what changes caused this and how they were resolved and how successful the solution was. IBM used DevOps based automation and implementation processes to optimize development processes [2]. DevOps approach was used in RFE (Requirement for Enhancement) community website where software users could submit improvement requests for software they purchased. RFE offers a way for users to take part in software development cycle thus facilitating user satisfaction and adding more value to the software in users perspective.

After RFE submission it is automatically synchronized to internal database where it is visible to suitable development teams [2]. Development teams are required to respond to new RFEs within 30 days. These are monitored with defined metrics used to calculate RFEs service level agreement (SLA). Formula used to calculate SLA is as follows:

$$\text{30 day SLA} = \frac{\text{RFEs Arrivals last 30 days} + \text{RFEs Need info and > 30 days old}}{\text{RFEs Arrivals last 30 days} + \text{RFEs in 30 day queue and > 30 days old and}} \%$$

When SLA of a certain product is over 85% it is considered to be in good standing. Case study was analyzed using these two data types: number of RFEs missed in 30 day period and development team's RFE SLA percentage.

Development teams struggled to give timely responses [2]. Two of the main reasons were not notifying project managers on RFE made against their product and burying opened RFEs to a large pool mixed with other requests. RFEs were hard to manage in pools with other requests and also their priority was hard to determine. Reports concerning RFEs were semi-manually generated by release manager. The report had to be requested from release manager and it took at least a half day to generate it. Report only provided SLA calculations and RFE status no further details concerning RFEs were provided.

Goal was to design a tool for automated and scalable reporting tool informing the development team through email [2]. Reporting tool process flow is depicted in figure 4. RFE data was pulled from the database with IBM Rational Software ClearQuest on a VM. The data was then analyzed by a java program generating the report. Report was then send through email server to different stakeholders including development and release management teams. Report process is timed to happen every regular business day mornings.

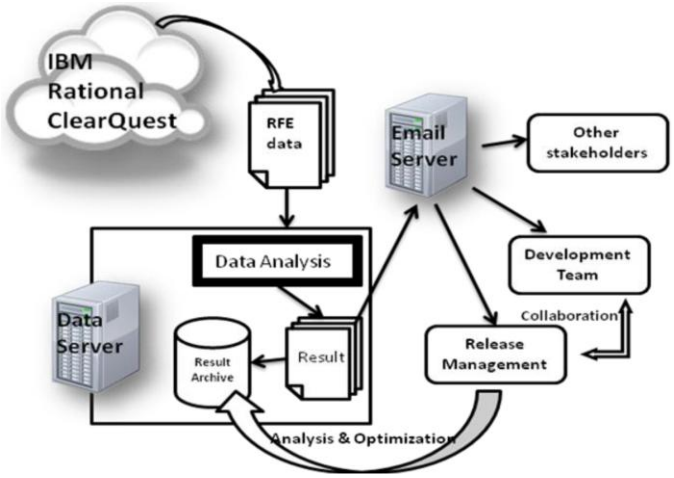

Figure 4. Reporting tool process flow designed in the IBM case study [2].

Before deployment the mean of RFEs missed 30 days threshold was 27,89 RFEs [2]. After deployment the mean dropped to 15,7 RFEs. Development teams average RFE SLA was before deployment 68% and after deployment it improved to 85%.

As it was analyzed in the case study paper this approach concerned only on one aspect of DevOps: improving delivery and feature flow [2]. Whereas monitoring and metrics were not taken into consideration. DevOps approach was only partially used. Based on the results of the case study the partial DevOps approach was deemed useful and it improved the development teams efficiency on RFEs. It was also stated in the case study paper that B2B (Business-to-business) is more competitive and the key to success is to manage development and delivery in an agile and continuous way. Thus closing the gap between customer expectation and product delivery is a key factor for large and small companies to stay competitive. Customer expectations being in line with customer satisfaction.

Future designs were determined from the case study [2]. Continuous interaction and feedback from all stakeholders is a key factor in allowing rapid delivery. Feedback has to be informative and there is a clear need for real-time monitoring on current processes. System transparency allows continuous verification and optimization.

VII. DISCUSSION AND COMMENTS

Using the DevOps approach improves product delivery and management thus creating value to the customer by extending agile software development principles and methodologies scope outside development to operations.

DevOps definition of development and operations combined from delivery and infrastructure perspective. It also mixes agile software development with Lean principles taking also

the business side into consideration [14]. Extending agile principles scope from development to whole system is beneficial to the business side. While DevOps is also very tool centered they should also be included in the definition. It is not clear whether this definition is targets a new group of IT-specialists or extends old IT job roles [3]. Still DevOps principles encourage the use of cross-functional teams and giving team members a more holistic view of the product. It also encourages developers to increase their skill sets across operations.

DevOps value-driven characteristics can be encapsulated to three statements: continuous delivery, more-stable infrastructure and amplified feedback [7]. Continuous delivery and amplified feedback enforces customer-driven development. Thus creating more value from customers perspective. This depends on how aware the customer is on market situation.

Continuous delivery depends on DevOps principles, methodologies and tools for making delivery process automatic [15]. DevOps itself is not the sole factor in the process chain but it tries to create a clear connection between them. Automatic processes deal with building, testing, deploying and version management. Testing prevents broken builds and sets a certain standard on what features and fixes end up in production. New features and fixes can be delivered more often to the customer thus adding more value to the product. Possibly enabling greater market share and new business ventures.

For continuous delivery to work in practice it requires more-stable infrastructure [1]. More time has to be spent in developing new features than fixing and maintaining deployment. Using DevOps tools and IaC principles delivery process can be made more stable and automatic. Aside technological needs continuous delivery also poses requirements on organizational structure. Teams should have clear goals and responsibilities which are more difficult to determine when dealing with cross-functional teams.

DevOps tools are centered on linux-based systems. Tools are not as universal as one would hope for but they are slowly extending their portability on windows-based systems. Of course one can use VM to run suitable operating systems but it can have some negative effects on performance. Cloud-based services are rather new technologies thus longevity and scalability are not always known making large company or enterprise level decisions harder.

IBM case study on partial DevOps approach gives some sample on how DevOps effects on development and communications. It also makes clear that even large companies can benefit from DevOps approach [2]. This however was only partial implementation and it is hard to determine whether results would be as good when all DevOps principles were to be implemented. It is true that smaller companies depend on rapid feedback and delivery however extending same ideas to a larger company is challenging. Also when viewed from a large organizations perspective the case study is on a small scale not giving a clear picture on how DevOps principles would affect the organization. Coordinating between many departments arises new problems to resolve. Careful analyzation should be made on limiting DevOps influence on organizational structures.

DevOps has clear advantages when compared with agile software development and emphasizes heavily on rapid and increased value delivery. Agile software development is required for DevOps to be implemented correctly. Extending agile principles outside development makes both development and operations more value-driven and customer centered [3]. DevOps has clearly sprung from the need of agile delivery management. Thus it is hard to distinct DevOps from agile principles and methodologies. It is easier to imagine DevOps as an additional layer on top of agile.

VIII. Conclusion

DevOps emphasizes on value-driven development and operations. Managing customer requests and value-streams it adds value to the product and makes value creation easier by changing developers time allocation from fixing and maintaining to development.

DevOps approach has technical and business benefits. Technical requirements concentrate on automatization and infrastructure. IaC is one of the main principles of DevOps. Automatic processes enable continuous software delivery. This approach also makes resolving problems faster and it also narrows down the scope of problems. Developers spend less time on fixing thus changing time allocation to emphasize development. All of these factors increase value delivery to the customer and making it faster. DevOps enforces competitive aspects and makes development and operations more value-driven and customer centric for both large and small companies.

DevOps combines development and operations by extending agile methodologies to include operations and systems. It delivers value by deploying new features and bug fixes more rapidly and improving infrastructure stability thus improving availability. Continuous delivery also amplifies the feedback loop between software provider and customer. DevOps methodologies are similar to agile methodologies but it has also introduced new methodologies such as VisibleOps which helps making processes more efficient and monitoring processes for identifying possible bottlenecks. DevOps tools help making infrastructure, monitoring and testing automatic. DevOps extends agile principles and methodologies making them more value-driven and customer centric.

Acknowledgment

No acknowledgments.

References

[1] M. Hütterman, DevOps for Developers, Springer science+Business Media, New York, USA, 2012.

[2] Y. Liu, C. Li, W. Liu, Integrated Solution for Timely Delivery of Customer Change Request: A Case Study of Using DevOps Approach, International Journal of U- and E-Service Science and Technology, vol. 7, No. 2, pp. 41-50, 2014.

[3] E. Mueller, J. Wickett, K. Gaekwad, P. Karayanev, What is DevOps?, http://theagileadmin.com/what-is-DevOps/, 26.10.2014.

[4] G. H. G. McDougall, T. Levesque, Customer satisfaction with services: putting perceived value into equation, Journal of Services Marketing, vol. 14, issue 5, pp. 392-410, 2000.

[5] J. Heikkilä, From supply to demand chain management: efficiency and customer satisfaction, Journal of Operations Management, vol. 20, pp. 747-767, 2002.

[6] K. Behr, G. Kim, G. Spafford, The Visible Ops Handbook: Starting ITIL in 4 Practical Steps, Information Technology Process Institute, 2005.

[7] G. Kim, The Three Ways: The Principles Underpinning DevOps, http://itrevolution.com/the-three-ways-principles-underpinning-DevOps/, 9.11.2014.

[8] P. Duvall, Breaking down barriers and reducing cycle times with DevOps and continuous delivery, (http://try.newrelic.com/rs/newrelic/images/GigaOm-Pro-Report-Breaking-down-barriers-and-reducing-cycle-times-with-DevOps-and-continuous-delivery.pdf) Gigaom Pro, 2012.

[9] Docker, What is Docker?. (https://www.docker.com/whatisdocker/, 7.12.2014)

[10] Vagrant, Why Vagrant?. (https://docs.vagrantup.com/, 7.12.2014)

[11] New Relic, Real-Time Analytics. (http://newrelic.com/solutions/real-time-analytics, 7.12.2014)

[12] Chef, How it works. (https://www.chef.io/chef/, 7.12.2014)

[13] Puppet, What is Puppet?. (http://puppetlabs.com/puppet/what-is-puppet, 7.12.2014)

[14] S. Sharma, Applying Lean and DevOps for better business outcomes, IBM: Invisible Thread, 23.5.2014. (https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/lean_assessment?lang=en)

[15] New Relic, The DevOps Lifecycle: Keep C.A.L.M. and Carry On. (http://newrelic.com/devops/lifecycle, 8.12.2014)

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI