HAAGA-HELIA
University of Applied Sciences

Software Development with DevOps

Rögnvaldur Kristinsson

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2015

HAAGA-HELIA
University of Applied Sciences

Abstract

27.5.2015

Author
Rögnvaldur Kristinsson

Degree programme
Business Information technology

Thesis Title Software Development with DevOps	**Number of pages and appendix pages** 34

The goal of the study was to introduce DevOps in software development, its methods and approaches to software development. I was introduced to the subject by one of my teachers, which let me into further studies and researched on the subject and finally choosing it for the study. The subject was completely new to me when I started the study which I found inspirational as I was learning a new and an interesting subject on a daily basis during the process of writing the study.

The material for the thesis comes mainly from books on DevOps, Lean and agile, as well as literature on Scrum and Kanban. There has been a limited number of internet sources used, such as articles, as well.

The thesis introduces the meaning and values of DevOps, and it´s importance in software development. In order to show the effect of DevOps in use, an example software development project was set up to help explain how DevOps could be beneficial and how it could improve the software development processes.

The thesis explains why methods such as those used in DevOps are needed and touches on the history of DevOps from both Lean and agile development perspectives.

The study concludes by analysing how DevOps can be beneficial to software developers and the author tests how DevOps works in a small-scale test project. During the conclusion the author goes through his own experiences and thoughts on DevOps and how it contributed to the author's personal and professional growth.

Keywords
DevOps, Lean, Agile, software development, development, operations, Java, Android

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

27.5.2015

Tekijä Rögnvaldur Kristinsson	
Koulutusohjelma Tietojenkäsittelyn koulutus	
Opinnäytetyön otsikko Ohjelmistokehitys DevOpsillä.	**Sivu- ja liitesivumäärä** 34

Tutkimuksen tavoitteena oli esitellä DevOps-menetelmät ohjelmistokehityksessä, aiheidean sain aluksi omalta opettajalta. DevOps oli minulle aikaisemmin tuntematon aihe joka vaikutti minua positiivisesti tutkimuksen aikana. Uuden aiheen oppiminen tutkimuksen aikana oli hyvä inspiraatio oman oppimisen kannalta.

Tutkimuksen lähteet ovat kirjoista, joissa käsitellään DevOpsin ja Lean-menettelyä sekä Scrum-viitekehystä ja tuotannon ajoitusjärjestelmää nimeltä Kanban. Internet-lähteitä kuten artikkelit on käytetty jonkin verran.

Tutkimus käsittelee DevOps-menetelmiä, niiden alkuperää ja tarkoitusta projekteissa sekä miten niitä sovelletaan ohjelmistokehityksessä. DevOpsin hyödyllisyyttä ohjelmistokehitysprojektiin ja sen prosesseihin käsitellään opinnäytetyössä esiintyvän esimerkkiprojektin avulla.

Opinnäytetyö kertoo, miksi DevOpsin kaltaisia metodeja tarvitaan sekä DevOpsin historiasta Lean- ja ketterän kehityksen näkökulmasta.

Tutkimuksen kirjoittaja käy työssään läpi omat ajatukset ja kokemukset DevOpsista sekä ketterien menetelmien käytöstä ohjelmistokehityksessä. Johtopäätöksessä kirjoittaja käy läpi DevOpsin vahvuudet ja puutteet käytännössä ja miten ohjelmistokehittäjät pystyvät hyödyntämään DevOpsista omista työtehtävistä. Tutkija myös kertoo, mitä itse on oppinut näistä aiheista tutkimuksen aikana ja miten se on vaikuttanut hänen oma henkilökohtainen ja ammattilainen kasvutta.

Asiasanat
DevOps, Lean, Agile, software development, development, operations, Java, Android

Contents

1 Preface

In its essence, software is a collection of pre-programmed instructions that allows users to communicate with a program and have it perform tasks and services that we require. Hardware can be described as the body of a device, but it is the software that allows the device to think and perform. Modern western society is heavily relied on technology, and there is software constantly at work all around us. Being so reliable on software for every day functions, work and entertainment it would be prudent to assume that there are fast and efficient ways of producing and implementing software to provide the best possible services to consumers.

That however is not always the case. The cycle of software development is not overly complicated. As according to the Waterfall model, the idea of software development is that there are a number of phases that must each be completed before the process can move on to the next phase. The Waterfall model was a breakthrough at the time, providing a clear and linear software developing cycle. The model left a lot to be desired however and the search for a better or the best software development model has been an ongoing process inside the development community. In 2009 the first devopdays.org conference was held and the term DevOps came into existing. DevOps has a number of different definitions and approaches, this thesis will focus on only one of them, which is the developer's perspective.

DevOps in its essence is very different from the waterfall model in the way that the Waterfall model was designed to optimize and improve the development process by focusing on the process itself. DevOps does the same thing and takes it further by improving the process, but the main focus of DevOps is the human factor in the development process.

DevOps represents strong and effective software development methods, but the core of the development successes lies in the people doing the development and DevOps focuses effectively on the human part.

If DevOps was a human I would say that the DevOps methods are the body and the limbs, while the humans putting the methods into action are the brain, the heart and the veins going through the body. That is the way DevOps works. Put effective methods and development cycles in place, while focusing on the people assigned to the process and continuously focus on and improve their cooperation and effectiveness.

This goal of this thesis is to open up the world of DevOps, explain its origin, capabilities and the tools it can use. The first part is more theory oriented to get the reader up to speed on DevOps. The second part focuses on DevOps in action. I constructed an example software application, to see how and if DevOps can be used in a small project like my own.

DevOps has gathered a huge following of supporters and in the process DevOps has given birth to new terms and ways of doing things. DevOps terminology will be explained as it is used in the thesis. Before I started on my thesis I was not acquainted with the world of DevOps in any way. It was suggested to me by a teacher, as a way to improve my thesis idea which was software development for Android in Java. At first I was hesitant to take on DevOps as it did not seem to fit my interests.

That changed very fast. After I started reading books, articles, internet discussions and magazines about DevOps I was deep in it. It gave me a different view on software development and broadened my horizon. I changed my thesis idea to focus much more on DevOps role and effects on software development rather than focusing largely on the software itself.

I started to look up other books about Lean and agile to understand better the background of DevOps and the ideas behind it. After reading a number of books, articles and professionals blogs on DevOps I started to plan and write this thesis.

2 The Definition on DevOps

When it comes to DevOps one of the hardest parts is to actually understand what it is and what it means. DevOps as a term is heavily and widely used in software development, and not always properly or correctly. Due to the nature of software development and how teams can have different needs and set goals for themselves, DevOps has been adopted to suit different functions and settings between groups and companies. That leaves the definition of DevOps somewhat open to interpretation by whoever is using it at any current time (Hüttermann 2012, 3-4).

What can be agreed upon by all DevOps users is that according to Sharma (2014, 3), DevOps in its essence is a mix of well-known, advanced practices and new, innovative approaches to common challenges in project life software delivery and operations. The fact that DevOps is so flexible and non-rigid in its interpretation and implementation is one of its strongest features as teams and companies can quickly and effectively adopt it to purely suit their group, while looking at other team's usage of it for guidance.

2.1 The Origin of DevOps

DevOps evolved from the agile movement, which in turn developed from Lean thinking. Lean is not a new form of thinking and it came to be long before software development did. In fact Lean thinking has been around for the last century or so. Lean thinking in its essence is reduction of waste, or removing unwanted items or unnecessary part of a process to make it more efficient and productive. Lean was simply adopted into software development. The most known part of Lean that is used in software development processes is the Kanban board which is actually taken directly from Lean. To describe Lean in software development, it could be said to be a complementary methodology incorporated into Scrum to control a software development cycle (Pugh 2010, 15-16).

The agile movement, originated from a group of people that came together to look for a way to remove the wall that stands between development and operations. The lack of cooperation between development and operations was a common occurrence back then, but few if any practical solutions had been introduced to fix the issue. The agile movement gave birth to the agile manifesto, a formal proclamation of four key values and 12 principles to guide an iterative and people-centric approach to software development. The agile manifesto seen in figure 1 below was coined in 2001 in the United States at a meeting solely held by IT professionals to find a common ground and try to find and alternative for a better more effective software development process as described in figure 1. The result was the agile manifesto and the agile movement was born (Shore & Warden 2008, 9-10).

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over Processes and tools

Working software over Comprehensive documentation

Customer collaboration over Contract negotiation

Responding to change over Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	*James Grenning*	*Robert C. Martin*
Mike Beedle	*Jim Highsmith*	*Steve Mellor*
Arie van Bennekum	*Andrew Hunt*	*Ken Schwaber*
Alistair Cockburn	*Ron Jeffries*	*Jeff Sutherland*
Ward Cunningham	*Jon Kern*	*Dave Thomas*
Martin Fowler	*Brian Marick* . (Shore & Warden 2008, 10.)	

Figure 1. The Agile Manifesto from 2001

The writing of the agile manifesto started the agile movement, which quickly gathered followers from all areas of IT development. Disagreements and debates about how to resolve the issues the agile manifesto aimed to fix continued, but there was finally a common understanding that better ways where needed for the future of software development. During the years that followed the agile manifesto, DevOps was slowly born. The movement itself still continues, and continues with the debate of better future for software development and Cooperation (Shore & Warden 2008, 11-12).

2.2 DevOps and its principles

The term DevOps stands for development and operation, indicating the two sections of software development. The term DevOps was coined over time, as people looked for better ways for developers and operations to work together. It was first used by John Allspaw at a conference in 2009, where he gave a lecture called "*Dev and Ops Co-operation*". The term DevOps stuck from then on, and articles, books and reports with the term DevOps started to appear soon thereafter (Hüttermann 2012, 4).

The principles of DevOps according to Hüttermann (2012, 4) come from the same core that brought DevOps into existence. DevOps strives to combine those principles into the DevOps process.
The main principles of DevOps are:

- Iterative.
- Continuous.
- Automated.
- Flexible.
- Collaborative.
- Holistic thinking.

While teams can have a different approach and implementation of DevOps to their projects, all teams should analyze their existing processes and compare them to DevOps principles to learn where and how DevOps can improve their own process and add value to it.

2.3 DevOps Values

The values and benefits of DevOps are threefold. There are business values, human values and customer values. All can benefit a company tremendously when properly executed. The business and customer values focus mainly on the financial benefits from DevOps, while human values focus more on human values on the personal level within the company and its teams (Sharma 2014, 4).

2.3.1 Business Values

For companies that are willing to take the plunge into DevOps and change the way they handle software development, there is a number of benefits to be reaped from the process. When DevOps methods are firmly in place, they allow the company to quickly maximize the speed of development processes or services, faster delivery to the customer and better and more reliable maintenance of their software.

IBM for example mentions in its own book on DevOps, “DevOps for Dummies”, that there are three areas where companies can benefit greatly from following DevOps practices. The first one is better customer service; by bringing a better service that has the customer more involved on all levels creates better ties and loyalty with the customer. For it to work, the company must continuously work on getting up to date feedback from all participants on the product, be it new software or an existing one (Sharma 2014, 4-8).

This makes sure that the customer always gets up to date information on the software and the development process. All participants have to be included in the process, be they users, targeted companies, suppliers and so on. If one or more participant is left out of the process, the process is more likely to fail, which in turn will affect the customers view on the project and the company. The second value is increased time to innovate. With better communications and organization and planning come new opportunities to improve old methods and bring about new ones. Clear thinking and planning allows for a much better usage of resources, and the extra resources that become available can be used for innovation and higher value activities.

The third and the last value is a faster time to value. DevOps provides ways of bringing software faster through development to production. While the value of the software of the development process can wary from company to company, the goal of DevOps if used as a business capability is to deliver value faster and more efficiently than before. (Pixton, Gibson, Nickolaisen 2014, 87-92.)

2.3.2 Human values

DevOps as seen above can be applied to business needs as well as IT needs. DevOps values however, are more focused on interpersonal aspects as DevOps tends to be a human problem. That is, the problem is not the software or the teams developing it. It is the people that make up the teams and organizations that are the problem. DevOps strives to bring about change and efficiency on many levels but it is the human factor that is the most important.

DevOps focuses not just on collaboration and trust; it's about making work and development fun and interesting for all involved. This means that in order to work, incentives and goals have to be the same for all the persons involved. As long as there are conflicting goals between teams there will be conflicts on all levels of development not including the strain it can put on the communications between the teams.

Getting everyone in both teams on the same page is critical. Respect for one another, understanding of the other teams work combined with shared values, goals and the feeling of collective ownership will create a powerful development group. This can effectively end the rift and conflict between the two teams and start a much more efficient and productive development process where both teams understand each other's role and functions in the development process. Every member of the development and operation has to be included. That means programmers, testers, QA and operations. Basically the teams become cross-functional and are all rewarded equally for developing a lot of stable and functional software. (Pugh 2010, 19-22.)

2.3.3 Customer Values

The customer values in DevOps are twofold. Firstly there are the benefits the developer gets in form of feedbacks from the customers and the value the customer gets for being able to participate in the development process of software. The customer brings a lot of feedback, user experiences, and their own ideas and opinions of the software to the developer, which otherwise the developer would not have access to. In modern software development it has become normal to rely on extensive user testing by potential customers who bring valuable information to the development process.

The customers themselves have become used to having access to the software during testing periods and a software developer not willing to allow customers to be a part of the process might find himself shunned by the customers as the software is ready. Customers are no longer willing to simply buy software or a computer game without having some sort of access to it or getting feedback from other customers as they test the software before buying it.

Customer's value in an age of social media is invaluable as positive or negative feedback can quickly spread to all corners of the world through applications like Facebook and YouTube. As such, it can be said that the customer has the upper hand during the software development process due to the huge impact their appreciation or unhappiness can have on software or a company. It is no longer the users following the developers; it is the developers following the users in a highly competitive digital world. (Hüttermann 2012, 70-72.)

2.4 DevOps Practices

To follow DevOps methods, the practices followed during a development process need to be in line with DevOps methods. Making sure that all relevant parties are involved in the process goes without question. This means that not only are all parties constantly aware of the current situation but that when an issue arises that needs quick attention it can be addressed and solved quickly without lengthy explanations to multiple parties.

Automated Testing is vital in DevOps. In DevOps testing many times a day keeps the code clear and possible bugs are found and dealt with early instead of dealing with them during a much larger testing phase. Change management is very important in DevOps where integrated change management means that the evolution of the IT infrastructure as whole goes smoothly for all parties involved.

This can be a major change to teams not used to integrated management on a large scale outside the development team. Continuous deployment, support and observation are DevOps way of constantly providing new updates, product support and observation of the entire software development lifecycle. DevOps focuses on small but frequent updates, close and fast support as well as a clear picture of the overall development process from start to support. (Pugh 2010, 23-24.)

3 DevOps in Software Development

3.1 Development and Operations Departments

Typical software developing departments and organizations divide their work force into different departments. This is due to the fact that when it comes to developing and releasing software, more is needed than just coding skills. Unless it´s a small company with only one team, there is always a development department and an operation department. Both are an important part of the software development process although their approach towards the software is different.

The development department´s function is to create code and updates to new or existing projects and hand them over to operations once they have been tested. The operation department then handles the release of the new software or the update to an existing one. Operations handle maintenance and system functionality as well.
(Hüttermann 2012, 17.)

The basic roles of development and operations in software development are separated and shown in figure 2 below as according to Hüttermann *(*2012, 5-8*)* and explained further by the roles of development and operations according Hüttermann (2012, 5-10). The purpose of the figure is to give a small insight into what kind of technical aspects each role has.

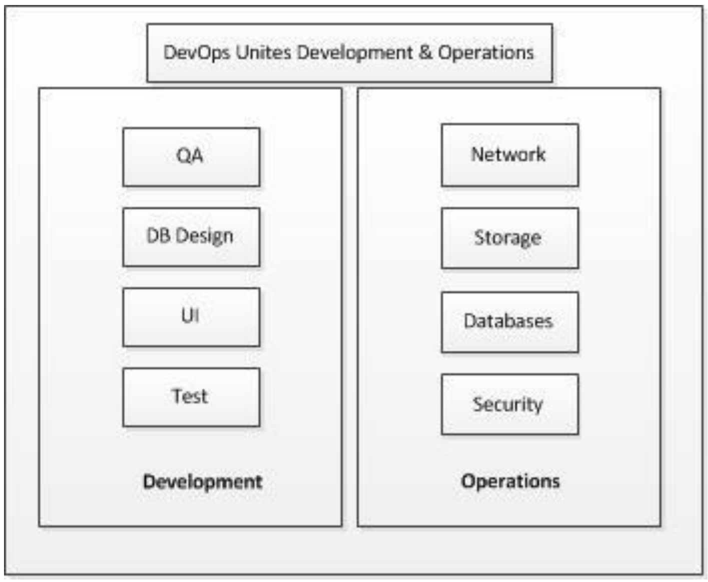


Figure 2. Functions of Development and Operations in Action.

Development:

- QA: Quality assurance role is to continuously make sure that the software fulfills the required quality demands.
- DB Design: The design and creation of all necessary databases for the software.
- UI: The user interface needed for the software. User interface needs to be clear, provide easy access to items and needs to be appealing to the user for it to be successful.
- Test: Testing is a part of the core of development. Continuous testing is necessary to find possible bugs and errors quickly and making the launching of the software a smooth and successful operation.

Operations:

- Network: For DevOps to work, the network system that the teams use for the development needs to be fast, effective and setup to suit DevOps development and operations teamwork.

- Storage: Storage is a critical factor as it is critical for DevOps teams to have all the necessary storage for their developing and operation tasks
- Databases: Databases in DevOps need to be well organized and organized for the teamwork between development and operations. Database design and function needs to answer the needs of both development and operations.
- Security: Security is critical in all software development. More so in DevOps as the process of DevOps requires many more parties involved in the process as before. Therefore it is extremely important that security is well thought and designed to prevent security leaks and data losses.

3.2 Business partners and suppliers in DevOps Lifecycle

Business partners play a much bigger role in DevOps than they would in a development process without DevOps mentality and planning. Before businesses would only see the product when finished, there was little or no interaction during development processes with the business partners and suppliers. With DevOps this has changed completely. Business partners and suppliers play and active role in the development process which means that they will have a much larger effect on the final outcome and there are less changes of disapprovals from the business side with the final product. Both business partners and suppliers do have a huge interest in the product as the financial implications can be significant.

Nowadays businesses are less willing to invest in projects and software without having any saying in the development and the final outcome. Having businesses and suppliers involved means that they constantly aware of the current situation and when necessary they can adjust their plans and expectations. From the supplier point of view, knowing exactly what to expect and to be able to make its own voice heard in the process makes it easier to optimally plan their business plans to be as effective as possible when the final product is ready. After the product has been released, the continuity of DevOps in a supportive and observing role allows the business owners to quickly adjust their business policies according to the functionality of the product and feedback from the DevOps operations team. (Sharma 2014, 20-22.)

3.3 Developing Software with Customers Using DevOps Oriented Development

The customers are the critical part in the development process as they will end up paying for the product, so getting constant feedback during the development is of the upmost importance. For the customer to be able to participate in the development process means that the developer cares about the customer's opinion and is willing to make changes to its product if necessary according to the customer's feedback.

Computer game manufacturers especially rely heavily on their customers to play their products in alpha and beta stages to get the best and most honest feedback as possible. Recently a number of game developers have decided to charge the users for becoming a part of the development process as it has become extremely popular. While charging users to provide feedback and testing the product may sound extreme, it does mean that the users that will pay to become a part of the development process will be very motivated and keen on providing the best possible feedback to the manufacturer.

3.4 People in Conflict

The very nature of the developers and operations makes clashes unavoidable if there is not proper cooperation and communication between the two teams. This difference and the failure to address this difference properly can bring a software development project into a quagmire and initiate the so called blaming game.

The blaming game is a direct result of the lack of communication between the teams. As an example, the development has coded and successfully tested a big update to existing software. The team hands the code over to operations that are tasked with getting the code into production. Operations then go ahead with releasing the code, which results in the crashing of the software in question. Operations try hard to fix the issue while receiving a lot of critic from the software users. Development also blames operations for the update was fully working during testing.

In the end the problem turns out to be either a bug missed in testing or a third party software that the update used has received and update which then caused the software to fail. This can lead to anger on the side of operations for being forced to publish a software update that caused the system to fail, affecting their work and reputation in the process. This brings about the fundamental difference and main area of conflict between development and operations.

Each department has an invested interest in its own part, both financially and professionally. The financial benefits for development usually come in the form of bonuses if development manages to release a certain amount of workable software in a month or a year. Operations on the other hand receive bonuses for keeping the systems working optimally and minimizing errors and bugs in the system. This clearly means that development wants to continually push forward with new software and updates while operations is reluctant to apply new software and updates that can affect system stability. Especially large updates, which have the potentiality to cause all kinds of system errors, are not welcomed by operations.

This can lead to development pushing out code continuously but the code is not released by operations for fear of system failures. Operations can also start to do their own testing on the new release to make sure it really works. In the cause of errors the code is send back to development, which having tested the code themselves successfully previously causes them to react with anger towards operations for meddling in their affairs This type of blaming game can escalate to the point that the software development is non-functional. Meetings can turn into angry shouting matches as each team brings it´s supervisor onboard for the meetings.

When the trust between the two teams has so broken down, something drastic has to be done. The only viable solution at that point is to completely rebuild the trust from the ground up, which in some cases can prove to be impossible. These types of scenarios are far from uncommon in the corporate world, and tackling them can be hailed as one of DevOps great successes. (Hüttermann 2012, 18-22; Shore & Warden 2008, 99-102.)

4 Adopting DevOps

For a company to decide on a switch to DevOps does not necessary mean that it is having problems with its software development or that its teams are stuck in state of conflict or mistrust. Well off companies might want to change to DevOps to improve its values, increase innovation and build for the future by creating an even more efficient team. It will also be a lot easier for successful co-operating teams to do the switch do DevOps than for teams that have deep mistrust for one another.

Adopting DevOps will take some time and practice before it will be functioning optimally. Teams will need to get accustomed to the “one team” functionality by working a lot closer with operations and testers.

The customers and users have to be incorporated into the process as well. Before the change to DevOps process starts DevOps definitions or architecture has to be setup and agreed upon. It is important that both development and operation are a apart of this process to make sure both views are taking into account and to avoid any future conflict by making sure that both teams are in agreement on all aspects of the DevOps definitions. (Hüttermann 2012, 65-72; Sharma 2014, 9-13.)

4.1 Defining a Team

To bring the different teams together to form a DevOps team, it is important to setup a list of definition that fits both teams and address their concerns and interests. Both teams need to play an active role in this process and be able to confidently sign it afterwards. In the following list there are examples of definitions that should be included to improve the team’s understanding, appreciation and cooperation with each other.

- Determine share goals
- Set clear boundaries
- Define an optimal solution for a faster deployment that can benefit all
- Define down time, which team members can use to reflect and analyze the project
- Success is celebrated together, not separately, this brings the teams together on a personal level
- Acknowledge and appreciate the other teams work
- Define and use the same practices and tools across departments and teams
- improve understanding by asking questions and give feedback between teams
- Improve trust by inviting other teams members to sit in on Daily Scrums and meetings

These are just a few definitions and ideas on how to get the teams together and improve cooperation and understanding on all levels. For DevOps to work continuous integration and cooperation has to take place. Trust is easy to lose but hard to regain when lost. (Sharma 2014, 15-18.)

4.2 Define DevOps as a Process

The first thing to do before adopting DevOps is to define DevOps for the particular company. What is the DevOps development lifecycle, and how it is defined in the software development as according to Hüttermann (2012, 81-84).

We can define DevOps by using four main areas of activity:

- Design and revalue
- Develop and testing
- Deployment and support
- Observations and optimization

4.2.1 Design and Revalue

The design and revalue is heavily focused on the business aspects of DevOps, by designing and providing continuous business planning that is constantly valued and revalued from the business perspective.

Businesses receive and act quickly upon customer feedback and testing results in DevOps. Therefore the business has to be agile and be able to make changes to the development process or to the product depending on the feedback they get. In the modern world, where changes are rapid and failing to keep up with the competitors and customers demand can quickly spell disaster for companies.

For example the Finnish game company Rovio, that created angry birds, became overly confident after the success and fame it received after the angry bird mania that swept over the mobile phone game genre. Today Rovio has allegedly lost millions of loyal players by failing to keep up with the fast changes to the genre and the number of success games brought out by competitors. Recently Rovio has revealed ambitious plans to reach the top again, but it shows how important it is to keep in touch with the market and the costumers.

Successful companies have applied lean and agile methods through DevOps by clearly identifying the outcome beforehand. Starting small, and then continuously adapting and adjusting their process based on the customer feedback they get. As changes have to be rapid and resources need to be used optimally and efficiently, it is critical to apply the agile methods DevOps stands for in the beginning of the software development process. (Pixton, Gibson, Nickolaisen, 2014, 75-77.)

4.2.2 Development and Testing

Development and testing consists of cross-functional teams by combining all developing parties including no-technical parties like suppliers and business partners into a collaborative development. Collaborative development allows all parties to apply a common set of practices and work platform to create the best software possible. These

practices were born out of the agile movement as a solution to the lack of integration between various partners which often hampered or completely prevented software to be effectively brought to production.
Testing is an important part of collaborative development and refers to the software being continuously tested during all phases of the development process. This allows the development team to verify and validate the code they are writing and quickly spot and fix possible errors and bugs by continuously testing the code. This kind of testing has to be done, automatically and in production like environments to get the optimal feedback back to the development team (Pugh 2010, 23-24).

4.2.3 Deployment and Support

Deployment and support in DevOps are heavily integrated and the software is setup with the idea of continuous life cycle. That means that after the software has been released, it is in a state of continuous changes by receiving and releasing as often as possible small and easily applicable updates and fixes instead of large and cumbersome updates. This brings new updates and changes much faster to the customer and provides a lot of important feedback for the production team (Sharma 2014h, 23-28).

4.2.4 Observation and Optimization

The combination of observation and optimization focuses on observing the product after it has been released and optimize it through feedback from customers and business partners. Necessary updates and changes can then be applied quickly and efficiently which in turn keeps the customers happy as their concerns are responded too quickly.

The development team can re-revalue its methods and processes during the software development with the help of constant feedback and update them for future projects. This loop of observation and optimization is a vital component of DevOps as it is concerns the core of making companies more lean, agile and responsive to its partners and customers. Ignoring the customer feedback can quickly result in disgruntled customers and unhappy business partners that feel that their feedback and pointers are being ignored. That can quickly cost companies its customers who if unhappy, will quickly move onto different software more to their liking. (Sharma 2014, 20-23.)

5 Agile tools for DevOps

There is more to DevOps than just applying and using it´s methods. For DevOps to work perfectly, teams need to agree upon and use the right tools for the job. While there are a number of tools that can be used effectively with DevOps and each developer will have personal favorites it is important that the tool that are used are suited for everyone and everyone feels comfortable working with them.

Effective DevOps teams can possible get good result with substandard or outdated tools but the same team can get great results giving the right tools for the job. For the DevOps spirit to really shine, the teams should be allowed to decide which tools are the right tools for them. That will bring the team members closer together and create a team spirit as the whole team is involved in the planning and decision making process.

Teams need to agree on tools such as configuration tools and maintaining tools and how to manage workflows.

Below is a brief introduction of a few agile tools that have become increasingly popular in agile software development. DevOps does not have any official list of which tools to use, but the following tools are often mentioned in books and text written by DevOps and Agile developers.

5.1 Choosing the Right Tools

RANCID stands for "Really Awesome New Cisco confIg Differ". Basically it is not a configuration tool per se, but a collection of utilities that can be used to check and go through the user's configuration revision control. The point of RANCID is that as software grows in size, and the code is going through frequent changes and updates, it often causes configuration issues with devices and components needed to run the software. RANCID helps to control and manage all those configurations.

RANCID collects and saves all configurations for all the devices on the system where it has been installed. This can prove to be an enormous help for large and complicated software development projects that can cause a lot of configuration and compatibility issues along the way. When properly installed, it provides the user with relatively easy way to access and manage all device configurations on the system. (Jonathan Thurman 2014, 1-4.)

Puppet is somewhat similar configuration tool as RANCID although it goes about achieving its goals differently. Puppet like RANICD is all about controlling and managing devices and their configuration. Puppet does it by managing the servers. The desired device configurations are fed to Puppet in easy to read declarative language. The configurations are then stored in the tool and can be accessed when necessary. This saves a lot of time when configurations go out of sync or for some reason need to be restored. Puppy is then able to quickly restore the optimal configurations.

Puppet on its own it's a strong tool for DevOps, but there are a number of add-ons that can be added to Puppet to increase it functionality. For example scalable warehouse database, value lookup tool, cross-platform system profiling library ad well as a powerful framework that can be used to control and access all the components that have been selected. The creator of Puppet, Puppetlabs has not focused specially on tools and frameworks for software development exclusively but for those who are interested in DevOps development, Puppetlabs has a lot of tools and framework that can be adapted into DevOps. (Puppetlabs 2013, 16-22.)

Chef is a maintenance tool designed to allow maximum efficiency when working on infrastructure. Infrastructure is always complex to maintain efficiently as it grows and Chef is designed to turn infrastructure into code. As the infrastructure has been changed to code, it can be changed, tested, updated and worked on as a normal software code. For optimal performance Chef needs to be installed on every server and virtual server on a system. Chef stores configuration data and manages the infrastructure by using so called recipes.

A recipe is a piece of the infrastructure, a file for example and in the recipe is a detail description of the file as well how and where it is installed or should be installed to. Chef´s infrastructure basically consists of recipes which in turn consist of resources formed from building blocks. Chef routinely checks for updates on the servers where it is installed and automatically installs updates to its recipes when necessary. (John Ewart 2014, 11-14.)

There are many other tools available that can fulfil the needs of any DevOps team, and choosing which one is the most suitable for the teams in question should be of the upmost importance.

5.2 Managing Workflows

Managing workflows is the backbone of DevOps as for the DevOps process to be fluid and constantly evolving there has to be a clear management of the workflows within the teams. SCRUM and Kanban are the two main methods used in software development for workflow management.

Scrum should be well known to any developer, and while it is not meant specifically for DevOps exclusively it is an integrated part of software development. Thus, scrum is important to the DevOps development process.

Scrum actually fits into DevOps mentality really well because it is designed as a way for teams to work together to successfully develop software. What makes scrum so good with DevOps is that scrum focuses heavily on the human factor that is do important in DevOps. Scrum focuses on the human factor of the process, improving communications and allowing each member of the team to communicate his thoughts and ideas to the project.

Scrum is not an overly complex tool. Scrum provides and focuses on a few rules that sets up and environment which allows the team to effectively and openly communicate together and identify and solve problems that might arise during the development process.

Scrum focuses on how to set up both product and sprint backlogs and how to designate people for the roles. According to scrum there will be a product manager, who is responsible for the team and handles communication with the project owner. According to scrum the project owner can only be one person. The product owner has the ability to decide what goes into the product backlog and what the items priority will be. The main point is that the product owner has total control over the product backlog and nobody has the permission to change it without explicit permission from the product owner.

The sprint backlog allows teams to break the project down to a number of sprints, each which lasts for a certain period of time decided by the team. The sprint backlog is setup with items from the product backlog and in accordance with the priorities that the product owner has set for them. Daily scrum are meetings where team members go through their tasks and they can briefly explain their view or grievances on the tasks at hand. The daily scrum is an important meeting that is critical for the human factor in DevOps. (Stacia Viscardi 2013, 18-26.)

Kanban like scrum is intended to visualize and improve the workflow of a development process by creating a clear picture of what needs to be done, when and by whom. Kanban focuses on the flow of the development process and identifies and deals with bottlenecks as they appear.

An interesting aspect of Kanban is the board that is used. Kanban sues a situation board with all the tasks that need to be done. As a task evolves, it moves to the next part in the development process. For example form development to testing. Large areas such as development can be split into two categories, doing and done for example.

For optimal Kanban results there are up to five core practices, depending on the size of the team that are considered to be vital for the development process to success.

- Visualization of the workflow
- Limited work in progress
- Manage flow
- Make process policies more explicit
- Improve Collaboratively

Presently it seems that Kanban is the more popular tool among software developers. (Hammarberg & Sundén 2014, 8-12, 46-56.)

6 Adopting DevOps for the First Time

6.1 The associated parties and the Business Idea

To adhere to DevOps in a development process, all the necessary parties that need to be involved in the upcoming development process need to be clearly identified and brought together to be involved in the process of identifying the goals and functions of the software that is to be created.

The goal is to observe and analyze how the different parties come together and strife to achieve optimal results, and understand how development process can be improved and optimized by using DevOps.

As the teams plan to apply DevOps for the first time, they need to be clear which partners are related to the process:

- Owner
- User´s
- Supplier´s
- Developers
- Operations

When the application is finished and has been handed over to operations for deployment, the teams can analyze the development process and pinpoint the areas where DevOps has been particularly effective and how it has changed from previous practices.

The business idea behind the software is of vital importance as any software will be built to accommodate and fulfill that idea. All parties need to understand what that idea is to work together for an optimal solution to it. Having all parties together at this stage is vital to make sure that all parties clearly understand what they are expected to bring to the development process and how they avoid any unnecessary misunderstanding or conflicts of interest later on in the process.

6.1.2 Applying DevOps to the Development Process

In order to apply DevOps to a development process a DevOps lifecycle for the development process is needed to identify each stage in the development process. As mentioned earlier, the DevOps life cycle consist of three main phases:

Phase one:

- The Users - Suppliers - The Company - Development/Testing - Operations/Support

Phase two:

- Plan and Define - Develop and Test - Release and Support

Phase three:

- Observe and Optimize

In the figure below is an example DevOps lifecycle as pictured by Sharma and IBM. The figure is clear, and it follows DevOps guidelines perfectly. The figure is added to show how a large player like IBM approaches and visions its own lifecycles in DevOps development.

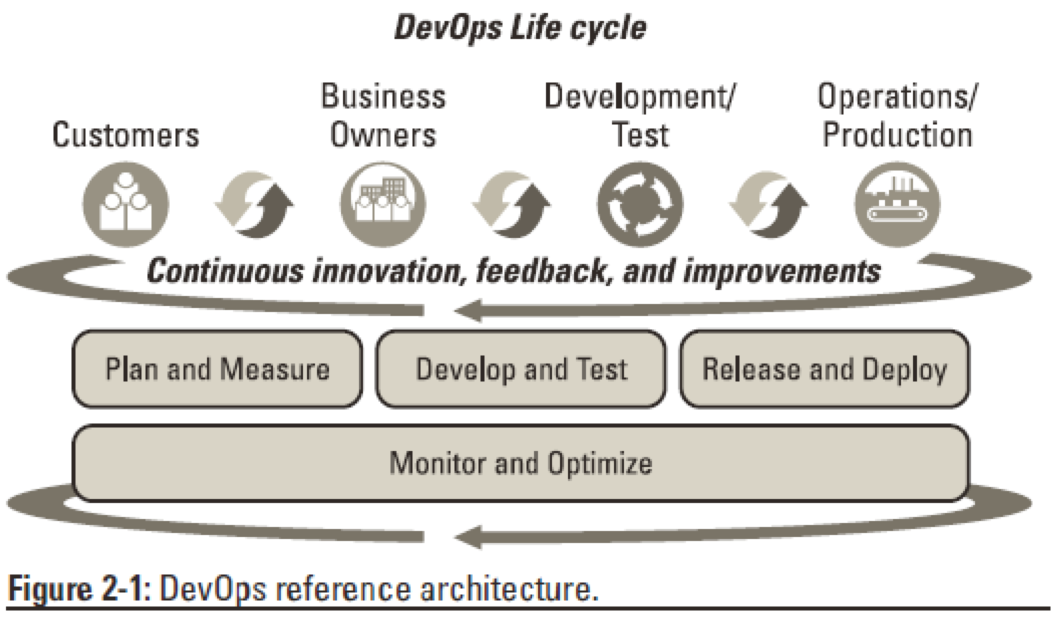


Figure 3. DevOps life cycle in IBM according to Sharma (2014, 10).

By setting a clear lifecycle for the software that will be deployed, it makes it much easier to stick to DevOps during the development process. After the idea for the software has been born, and the owner knows what he wants, he can setup a meeting with all the parties involved in the lifecycle and begin the process of how to define, design developing and maintain the software.

6.1.4 The Software development using DevOps

As the teams start the software lifecycle the real challenges of DevOps start to kick in. The teams need to integrate and apply DevOps to their cooperation which they may need time to get used to. As the software development progresses, the software teams keep getting feedback from the associated DevOps groups. With all the group´s constantly active and taking part in meetings, they have a clear view of the software and how it progresses. At this stage challenges can start to appear. The users testing the application may have issues with how the software looks or how it works. The supplier who supplies the necessary hardware or software may suddenly want to make changes to hardware or software versions that might force the developers to make changes to their work.

These issues need to be dealt with calmly and efficiently in the DevOps manner to avoid unnecessary conflicts and to find the optimal solution to each feedback. Development and operations have to take a clear line on all software related changes. Changing the look or do minor changes to the software should not be overly difficult, but larger changes can be. Changing the software to work on different hardware or operations systems that had not been agreed upon previously can cause a development process to stall. The optimal solution would be to continue with the software development, while including minor adjustments based on the users feedback. Larger changes like making the software compatible with new hardware or working on another operation systems would be rolled out in small but quickly released updates after the software has been released.

This will give the development and operations teams the flexibility to finish the software on time as well as set a clear update schedule for other hardware and operation systems. Instead of waiting a long time for a massive update that will support multiple operation systems, the teams will release a weekly or a monthly update that adds support for one more device or operation system.

Having the suppliers and the users as a in integrated part of the development process shows the strength of DevOps as each issue can be tackled and dealt with right away, instead of facing a lot of complaints from the users and the supplier who might feel that their views have been ignored. Dealing with the issues right away brings a stronger feeling of teamwork and cooperation and increases the trust between the different associated parties involved in the process.

As the software is ready to be released all parties know exactly what to expect, they know the features and limitations of the software and they know how development and operations are planning to expand the software in the future. The owner is happy that the software development process was a success and it can be expanded in the future. Furthermore the company has successfully started its first project using DevOps.

The software will continue its lifecycle through updates and changes. As the teamwork between development and operations improves, so does the quality and efficiency of their work. By working out a set update schedule for the software early in the development process the teams have overcome the biggest obstacle that development and operations teams face in software development, how and when to apply updates to existing software.

6.2 DevOps Effectiveness in a Small Project

While DevOps can undoubtedly have a major impact on midsized and large software development projects, I will explore what effect if any DevOps can have on a small scale project, a project on a more personal level. I will create small software from scratch and see how I can possibly benefit from what I have learned from DevOps during the development process.

I will create a small text editor application for my Samsung android tablet. I will design it, develop and test it and finally install it on my tablet. I will explain the technical aspects of the application, its functions and how it can be updated in the future. Finally I will analyze the development process and examine how DevOps has affected the development process. The application as mentioned is a text editor. It will be named NordSoft Notes. The editor will have three available functions. The user can add, change and remove text from it. As a test application it will have a very basic layout. The applications will not use data storage like SQLite and only store information locally as it’s a test application.

The operational system of the application is android 4.1.2. For simplicity´s sake, the app will not work on older android tablets as there are a lot of old android versions available which would make it harder to support. The coding and the framework will be done in Java, using Eclipse sdk for android development.

Android was originally developed by Google as an open source project and has become extremely popular due to its flexibility and the fact that it is free to use for software development. Android can be used to develop applications, games even complete operation systems. Due to the popularity of android there is a large amount of devices that run on android using a large amount of different versions.

Android devices are highly flexible and customizable and offer users the abilities to change the looks of their devices to fit their taste. There are a lot of different apps available to android both free and paid. Android updates are always free for the users although older android devices cannot necessary run the newest android versions.

Android development is different from development for IOS and Windows systems due to the large amount of different android versions available. IOS and Windows software are developed to work on their own devices running only on their operation systems.

Android software has to be compatible with a massive number of android devices running on multiple versions of android from many different manufactures. Due to this reason when developing in android it is vital to decide before hand which android versions the software will support and what kind of devices it is optimized for. Trying to develop android software that will work on all versions on android and all device sizes is a massive undertaking not to be taken lightly.

6.2.1 Android 4.1.2 and Java

Java is the programming language for android for all android devices. Java itself is an open source language so users that want to create their own software, can do so using java android free of charge. Android 4.1.2 is still fairly popular although not the newest android version out there. The newest version is android 5.0 or Lollipop. The 4.1.2 version was chosen because it is very stable and in use worldwide on many devices. There are a lot of older versions still being used as well, but as mentioned earlier the 4.1.2 version will be the oldest version supported as well as the focus version for the software.

Java is a free programming language that has been in use since 1995. As a programming language Java gets much of the syntax it uses from Microsoft´s C and C++ programming languages. Java is extremely popular language and a lot of web services will not work unless the user has Java installed. Java is great for programming for android devices as it is very fast, secure and reliable. In the modern world of technology java is virtually everywhere. To be able to develop java applications for android it is necessary have a version of Eclipse IDE installed with java sdk manager for android development. The sdk manager comes with an android virtual device manager witch is essentially a java virtual device that can be run from eclipse.

The virtual device created has to match the version (4.1.2) and screen size of the device that the software will be deployed on. It is possible to create multiple emulators that can simulate different phones, tablets and android TV's.

The user environment is Samsung GALAXY Note 8.0 tablet. The tablet runs android 4.1.2 which is a couple of years old version but one which has proven it to be very stable on all devices. The software application will be transferred to the tablet when it works fully without errors in Eclipse emulator.

6.2.2 The Software Lifecycle

The software lifecycle is much simpler than in a large development process as there is only one active "group" in the development process.

Phase one:

- The User

Phase two:

- Plan and Define - Develop and Test - Release and Support

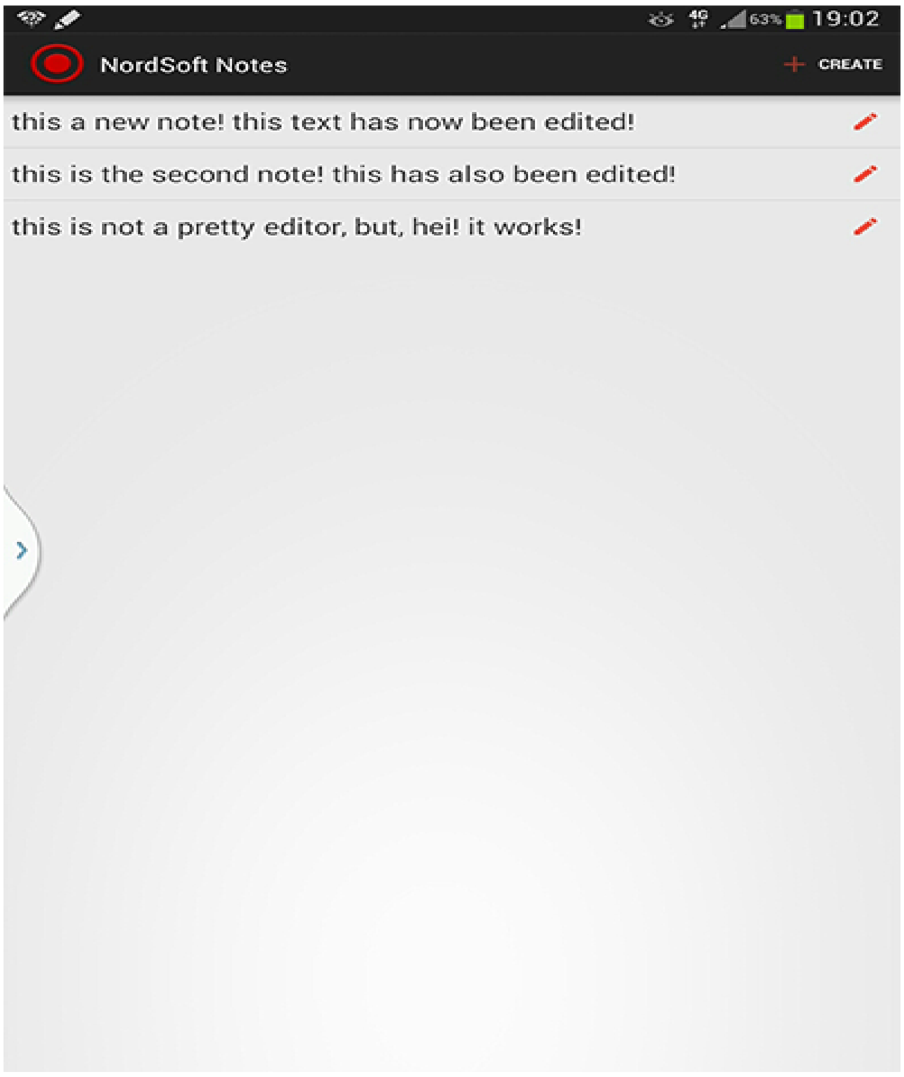
Phase three:

- Observe and Optimize

DevOps demands a lifecycle for its development processes. In this case however, as there is only one person or group active in the lifecycle it does feel different as the human factor has been removed. The lifecycle does however give clear picture of the development phases and gives a stronger sense of meaning and understanding to each phase.

6.2.3 The Application on a Tablet

The application has been successfully developed and installed on the tablet. The figure below is a screenshot of the application in action.



this a new note! this text has now been edited!

this is the second note! this has also been edited!

this is not a pretty editor, but, hei! it works!

Image 4. The application installed on a tablet.

7 Lessons Learned By using DevOps in a Project

The lessons learned by using DevOps in a small project are many. The main lesson is that the lessons learned from applying DevOps to large projects by improving cooperation between development and operations and including all parties in the software lifecycle, creating a sense of teamwork and a bond between the different parties is completely missing from a small project like my own. The human factor is the most important part of the DevOps mentality and in a small project like my own, DevOps cannot fulfill its main role as intended. The human factor was not wasted however, as I was able to take what I had learned from the human factor in DevOps and use it while I designed my own software. While DevOps is not designed for a one person development, a single developer can still learn and benefit from DevOps way of thinking and organization methods in his own work.

As a single developer, DevOps had a much larger impact on the technical factor instead. By creating a clear lifecycle for the development environment and setting up a clear update schedule in DevOps spirit, DevOps methods allowed for a better, clearer and more focused development process. As the human factors will greatly improve software design, development, testing and support, the technical side should not be underestimated. Following the DevOps lifecycle created an image that was easy to follow and it gave the overall process a feeling of clarity and understanding. Thus, DevOps does not only fix the issue between development and operations, but improves the technical factor as well.

8 Summary and Conclusions

This thesis has done a lot to introduce DevOps, its origins and purposes. There is still a lot more to DevOps as this thesis focused mainly on DevOps in software development with small side steps to the business side on times. The origin of DevOps from Lean and Agile has been introduced and the values of DevOps have been introduced and discussed. The implementation of DevOps has been discussed theoretically and introduced by setting up an exemplary software development with DevOps that had the job of briefly introducing how a company that had never used DevOps previously. The conflict and cooperation between development and operations has been one of the main points of this thesis as DevOps came into being as a way to improve the teamwork and understand between them and decrease conflicts and mistrust in software development projects.

I have personally learned a great deal about Lean, Agile and DevOps during the process of writing this thesis. As my understanding grew so did the scope of this thesis. As the thesis is at its end I see that the world of DevOps, Lean and Agile is a vast and fascinating world that I am surprised I had not noticed before in my studies. I will continue with reading and learning about agile development and DevOps in action. I find DevOps fascinating and often find myself thinking as I do both school project and projects that I do on my own time about how that particular project would work in DevOps and how would I implement it.

The small software that I created was designed to be my chance to learn something new while doing the thesis. Instead the software became a side step as I continuously found new books and texts online about agile development and DevOps. I was particularly interested in reading about how agile and DevOps methods have been successfully implemented in real life and the change it has brought to different teams and companies.

In the beginning of this thesis I was not overly motivated and considered it a chore rather than a chance to learn something new. That changed quickly and I can walk away from this work content with my own efforts and with the feeling that I have learned something new and exciting.

References

Ewart, J. 2014. Managing Windows Servers with Chef. Pack Publishing. Birmingham.

Hammerberg, M. & Sundén, J. 2014. Kanban in Action. Manning Publications Co. New York.

Hüttermann, M. 2012. DevOps for Developers. Apress. New York

Pixton, P. Gibson, P. & Nickolaisen N. 2014. THE AGILE CULTURE LEADING THROUGH TRUST AND OWNERSHIP. Pearson Education, Inc. Boston.

Pugh, K. 2011. LEAN-AGILE ACCEPTANCE TEST-DRIVEN DEVELOPMENT. Pearson Education, Inc. Boston.

Puppetlabs. 2013. WHAT IT IS & HOW TO GET STARTED. Puppetlabs.

Sharma, S. 2014. DevOps for Dummies. John Wiley & Sons, Inc. New Jersey.

Shore, J. & Warden S. 2008. The Art of Agile development. O'Reilly Media, Inc. California.

Thurman, J. 2014. 5 Unsung Tools of DevOps. O'Reilly Media, Inc. California.

Viscardi, S. 2013. The Professional ScrumMaster´s Handbook. Packt Publishing. Birmingham.