jamk.fi

Version Control System

Designing and Implementing Server Infrastructure

Riku Ojala

Bachelor's thesis
December 2016
Technology, Communication and Transport
Degree Programme in Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Ojala, Riku	Bachelor's thesis	09.12.2016
		Language of publication: English
	Number of pages 53	Permission for web publication: x

Title of publication

Version Control System

Designing and Implementing Server Infrastructure

Degree programme

Software Engineering

Supervisor(s)

Marko Rintamäki

Assigned by

Senop Oy, Optronics, Lievestuore

Abstract

Senop Oy Optronics R&D software development team had an outdated VSC server. Senop Oy assigned to set up and configure a server with VCS and supporting services for software development purposes.

The objective was to design and implement an improved server environment according to the requirement specification. The design included the server architecture, network topology, used software services and configuration of the services. First the basic setup for the server was done. On top of the basic setup, one VM was installed with a more advanced configuration of the services. The server, NAS server and developer PC's were connected to a switch allowing a network connection between them. The server and supporting services were implemented and tested according to the specification.

As a result, all the devices and services were set up according to the requirement specification. The server set up supports the software development work and the important data of the server is backed up in NAS server. Senop Oy R&D team were pleased with the server setup and it influenced the software development in the R&D team positively. The project started a continuous development process of server environment and software development methods.

Using VM's it is possible to create technically difficult and advanced setups without interfering with the current server environment. Using VM's allows to develop the server environment in stages and it can be tested outside the server environment. However, VM's tend to use plenty of system resources which may result in problems later. Alternatively to VM's one may use application containers to optimize system resources.

Keywords/tags (subjects)

Linux, Server, Version Control System, Virtualization

Miscellaneous

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisu laji	Päivämäärä
Ojala, Riku	Opinnäytetyö, AMK	09.12.2016
		Julkaisun kieli: English
	Number of pages 53	Verkkojulkaisulupa myönnetty: x

Työn nimi

Version Control System

Designing and Implementing Server Infrastructure

Tutkinto-ohjelma

Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)

Marko Rintamäki

Toimeksiantaja(t)

Senop Oy, Optroniikka, Lievestuore

Tiivistelmä

Senop Oy optroniikka T&K-ryhmän ohjelmistokehittäjien VCS-palvelin oli vanhentunut. Senop Oy:n toimeksiantona oli rakentaa ja konfiguroida palvelin VCS-toiminnolla ja ohjelmistokehitystä tukevilla palveluilla.

Tavoitteena oli suunnitella ja toteuttaa paranneltu palvelinympäristö vaatimusmäärittelyn mukaisesti. Suunnitteluun kuului palvelinarkkitehtuuri, verkkotopologia, käytettävät ohjelmistopalvelut sekä niiden konfigurointi. Ensin tehtiin palvelimen perusasennus. Perusasennuksen päälle asennettin yksi virtuaalikone, joka sisälsi edistyksellisemmän ohjelmistopalveluiden konfiguraation. Palvelin, verkkolevypalvelin ja ohjelmistokehittäjien koneet yhdistettiin kytkimeen, joka mahdollisti verkkoyhteyden laitteiden välilä. Palvelinympäristö testattiin, jotta se vastaa vaatimusmäärittelyä.

Lopputuloksena palvelinympäristö vastasi vaatimusmäärittelyä. Palvelinympäristö tukee ohjelmistokehitystyötä ja palvelimen tärkeät tiedot on varmuuskopioitu NAS palvelimelle. Toimeksiantaja oli tyytyväinen palvelinasennukseen, ja sillä oli positiivinen vaikutus ohjelmistokehitykseen T&K-ryhmässä. Projekti käynnisti jatkuvan palvelinympäristön ja ohjelmistokehityksen käytänteiden kehitysprosessin.

Virtuaalikoneita käyttämällä on mahdollista saavuttaa teknisesti haastavia ja edistyksellisisä ratkaisuja ilman, että nykyinen palvelinympäristö häiriintyy. Virtuaalikoneiden käyttö mahdollistaa palvelinympäristön kehityksen vaiheittain ja niiden testauksen palvelinympäristön ulkopuolella. Virtuaalikoneet kuitenkin käyttävät paljon järjestelmän resursseja, mikä voi tuottaa ongelman myöhemmässä vaiheessa. Vaihtoehtoisesti virtuaalikoneiden sijasta voisi käyttää sovelluskontteja järjestelmäresurssien optimoimiseksi.

Avainsanat (asiasanat)

Linux, palvelin, versiohallinta, virtualisointi

Muut tiedot

Content

Figures

Tables

1 Background and objectives of thesis

1.1 Senop Oy

Senop Oy is a company which develops and constructs reliable equipment and systems for safety- and security-critical applications. Senop Oy began operating at the beginning of year 2016, when Millog Optronics' Product Business line and Oricopa Oy's System Integration Business line were merged into the established company. Senop Oy develop and sell Millog Oy's existing products, including image intensifiers, night sights, thermal cameras and target acquisition and observation systems. Senop Oy's products and services also include the development, design and manufacture of mobile tactical systems, C4I-platforms, mid-life upgrade programs and system integration services. Products are designed to sustain fully operational state under extreme conditions and used by Defense Forces, Border Control, Special Forces, Navy, Police and Customs in several countries in a wide range of operations. (Patria Oyj 2016)

1.2 Thesis background and assignment

Senop Optronics R&D software development team have been using a server with Subversion service installed on it, to quickly allow it to act as a CVCS server. However, the server was poorly configured and it was running unsupported and obsolete operating system. Supported software for the operating system was very limited and it caused issues with the configurability. New security updates were not provided for the operating system which was considered as a security risk. The server was not configured with any backup solutions and the system backups were taken manually on a weekly basis by company workers using an external USB hard drive. This approach, however, added unnecessary workload for the workers.

To solve the problem, Senop assigned to set up and configure an improved server with VCS for software development purposes. The server consists of VCS and supporting services such as automated backup and logging service.

1.3 Thesis Objectives

The thesis objective is to design and implement a server environment which fixes the discovered flaws within the current setup. First the basic setup for the server is done with care. On top of the basic setup, VM's are installed which allows different environments and more advanced setups as needed. These VM's are configured and tested in other place before implementing to the server. As long the basic setup of the server environment is stable, it can be expanded and developed with VM's. This design allows continuous development and make the system very flexible.

1.4 Technical requirements

More detailed requirements specification was composed based on multiple conversations with company workers. The specification was reviewed and approved in the meeting. Technical requirements set in the specification were the following

- REQ01: Server MUST automatically backup files to Network Attached Storage (NAS) - frequently changing files once a day and full system backup once a month.
- REQ02: Every successful backup MUST be logged.
- REQ03: Server MUST maintain the integrity of the files.
- REQ04: Server files MUST be mirrored at least on two hard drives.
- REQ05: Server and services MUST be reached at normal working time in minimal delay.
- REQ06: Server configuration MUST be moveable to another system.
- REQ07: Server files MUST be able to recover after system failure.
- REQ08: Server MUST be able to recover from programmatic errors after user input.
- REQ09: Server MUST be able to detect and inform the user of possible failures.

- REQ10: Server network MUST be isolated from other networks.

2 Research method

2.1 Research Approach and Design

The thesis work is approached with qualitative Design research. The research approach contains planning, data collection and analysis which are the same as in qualitative research, however there are also actions which are performed after the analysis which aim at a change or improvement. The research approach is similar to Action research, however, it differs in participating in the operations of the development object which is element of non-social nature such as products, services, processes and actions. The objective is to produce functional and practical solutions that can be used in practice. (Kananen 2013, 40-49)

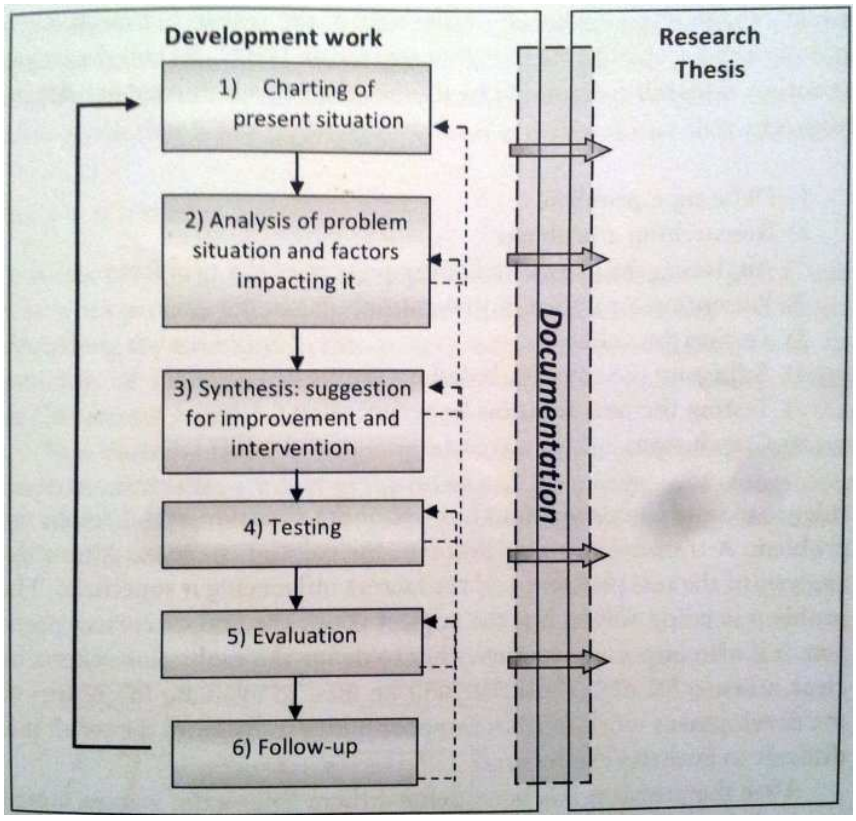

Figure 1. Stages of development cycle of design research (Source adapted from Kananen, J. 2009)

The research design follows the same steps as presented in Figure 1 where the research is illustrated as stages of a development cycle. First the present situation is well defined and the problem is determined. This way the objectives can be set and the context can be familiarizing with. After charting the situation, an analysis is carried out to find the root causes and factors which affect the problem. With the analysis, the design can be delimited and tested, followed with suggestion and intervention which is the implementation of the suggested design. The implemented design is tested according to earlier defined objectives that are tried to be solved and the test results are evaluated by measurements. The evaluation gives an idea of how successful the design and the implementation was. The evaluation process leads to a follow-up where the big picture of the development is reviewed and a new cycle can be started if necessary. The stages may proceed back and forth as needed if the next stage needs to fill up with more information. (Kananen 2013, 60-61).

If there is no exact measure for the effects of the change, it is evaluated with a story such as Most Significant Change (MSC) Technique. MSC-technique forms stories which show the change, which the change resulted from, why it is significant, when it happened and who were the actors included (Kananen 2013, 93).

2.2 Data Collection and Analysis

Data collection is divided into three phases. which are carried out before, during and after the design and implementation processes of the server.

The first phase of the data collection was conducted before the implementation process. Data was used to chart the current and the target state of the IT system to produce a requirement specification with design suggestion. Data was gathered through interviews with company employees.

The second phase of the data collection was carried out during the implementation process. Data was used to support the implementation process, and it consists of detailed information of how the different services are implemented in the IT system. Data was gathered through interviews with company employees who are mainly using the IT system, and using research material from the internet and books regarding to services which are designed to be used within the IT system.

The third phase of the data collection was completed after the implementation process. Data was used for evaluation and follow-up purposes. Data was gathered from user feedbacks and tests of the IT system. The evaluated information shows how the implemented design fulfilled the requirements, and if there are new suggestions for improvement which can be done with new iteration.

3 Theoretical background

3.1 Version Control

3.1.1 What is Version Control

Version control helps to manage file changes. Recording changes of one or more files over time allows specific version of the file to be reviewed later. Purpose of the Version Control is to provide clarity to when, why and what the contents change were. Importance of the Version Control grows in collaborative work where more than one person works with the same files. It is very important to know who changed the files, when the files were changed and why they were changed. These changes are collected and brought together into a software tool which is capable of recording and unifying all changes. This kind of tool is called as Version Control System (VCS).

3.1.2 Benefits of Version control

Version control in software development is an essential part of the every-day work. Without version control, collaborative work between several developers would be tedious as no one would know what changes are made and how the changes might conflict with each other.

VCS have change history of every file creation, modification and deletion. The changes are annotated with information of author, authors message describing the reason of the change and date. Use of the change history helps with long term design and work with legacy source code. VCS allows to revert files or even entire project back to a previous state and compare changes over time. This is useful especially in cases where some changes might be causing new problem and then it is necessary to recover the changes for deeper inspections.

With VCS, developers are able to keep track of all the modifications and follow the development work of each contributor which helps to prevent concurrent work from conflicting. For this, VCS offers ability to work on independent branches and allows developers to work without disturbing the main source code. Branches can be used for different purposes and there are different workflows to which developers can choose from.

3.2 Version Control System

3.2.1 Local Version Control Systems

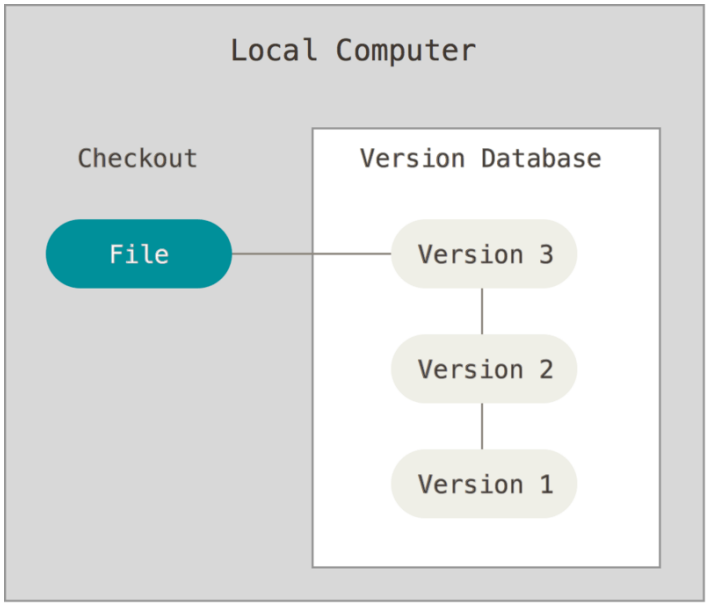

Figure 2. Local Version Control System (Source adapted from Chacon 2014).

Local Version Control System (LVCS) is one of simplest VCS methods available and it is run locally without a central server. It is useful in single-user scenarios where files are in dire maintenance. The problem with LVCS is the design which makes collaboration with others problematic. Also, there are no built-in tamper protection mechanisms and users who have access to the tools to version a file are also able to directly manipulate the corresponding version control file. (Chacon 2014, 27-28)

Examples of such tools are Revision Control System (RCS) and Source Code Control System (SCCS). To keep the concurrency, these VCSs rely to file locks. The file lock

allows the file to be edited only by one person at a time. LVCSs are considered as first generation of VCSs. (Version Control by Example)

3.2.2 Centralized Version Control Systems

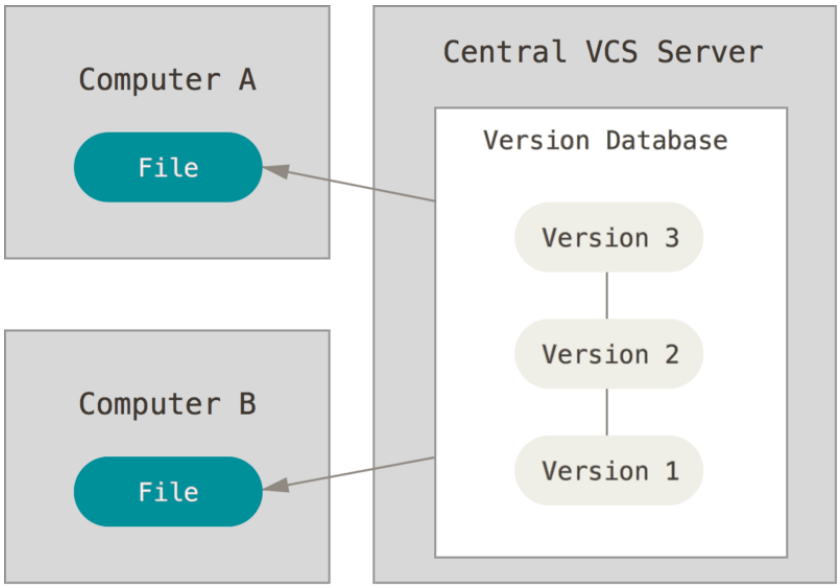

Figure 3. Centralized Version Control Systems (Source adapted from Chacon 2014)

Centralized Version Control System (CVCS) is a successor of LVCS and was created to enable multiple collaborators to work on projects together. These systems store all the information in a centralized server, and the version control operations of all collaborators must go through this server. All collaborators can download the current revision from this server, apply their changes and share changes among the others by committing them back to the server. Despite of the fact that the flaws of LVCS were taken out, new drawbacks came to scene. Due to the design, network access to central server is required for all version control operations, and the central server is a single point of failure. The network with a server adds administrative overhead and permission management must be configured and maintained to prevent unauthorized access. (Gyerik 2013, 13)

Example of such tools are Concurrent Versions Systems (CVS) and Subversion (SVN). To keep the concurrency, CVCSs rely on merge before commit procedure. This ensures the latest available source code from central repository is adapted and merged before the main code base is updated. Multiple persons are able to work

with the same code at the same time. CVCSs are considered second generations of VCS's. (Version Control by Example)

3.2.3 Distributed Version Control Systems

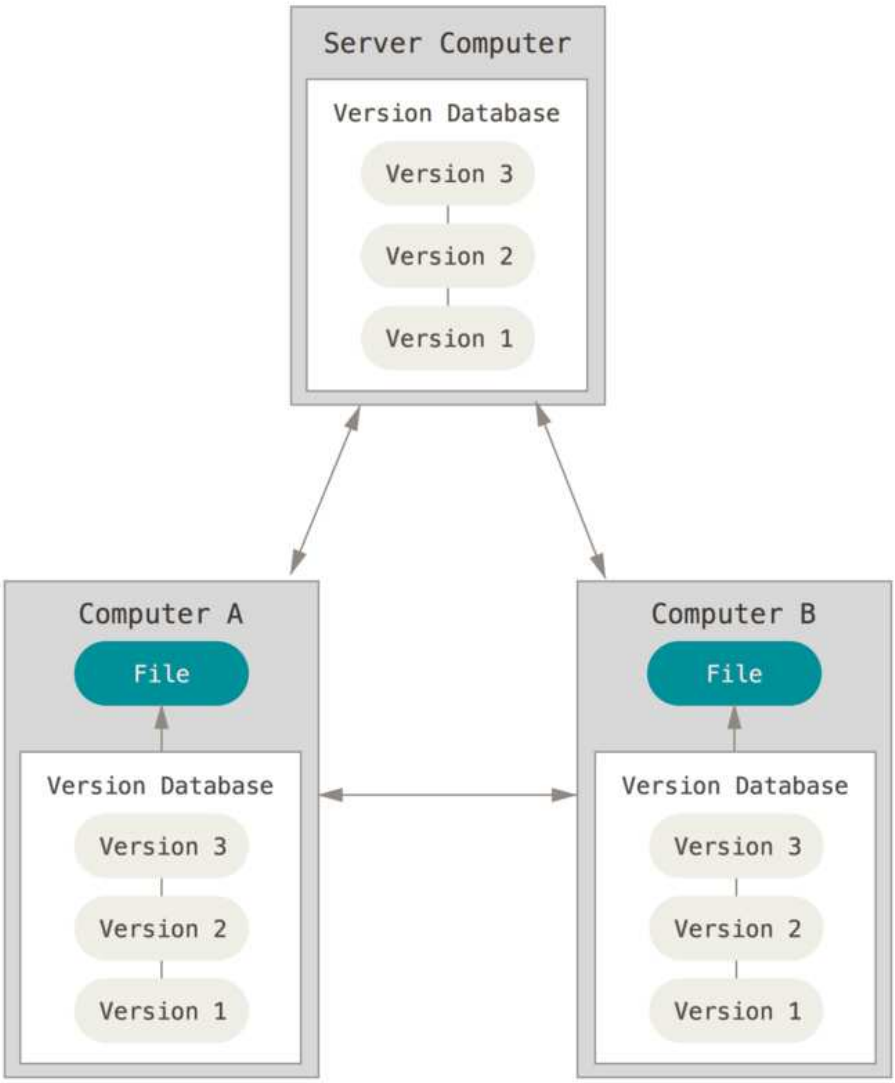

Figure 4. Distributed Version Control Systems (Source adapted from Chacon 2014)

Distributed Version Control Systems (DVCS) were created to make collaboration possible without a need for a central server. Instead of storing all the information and acting through centralized server, each collaborator has all the information and is able to commit changes locally to their own personal branches. Technically there is no need for a central server, however, most often there is a designated centralized branch aggregating the work of all collaborators. By design, it is very easy to replicate

the full revision history and it can only get lost if all the collaborators lose all their work. (Gyerik 2013, 14)

Examples of such tools are Bazaar, Git and Mercurial. To keep the concurrency, DVCS's rely on commit before merge procedure. DVCSs allow the merge and commit operation to be separated, which allows the users to make commit operations more as they work, and after the work is finished the changes can be merged back to the main code base which may be a decided central repository or the one responsible for the software. Multiple persons are able to work with the same code at the same time. DVCSs are considered third generation of VCSs. (Version Control by Example)

3.3 Version Control Processes

3.3.1 Reverting

VCS allows to revert files or the whole project to previous state. VCSs have a log of changes which contains a full history of the project. (Gyerik 2013, 8)

3.3.2 Logging

Log information is not optional as it is generated every time when changes are recorded. It includes a user-supplied description line, the date and the author's name. Typically, the description is at least a summary of changes optionally followed with more detailed information. (Gyerik 2013, 8-9)

3.3.3 Comparing

VCS allows to view the difference between two states of the file and it is most useful when comparing human readable text such as software source code, system script or other plaintext files. (Gyerik 2013, 9-10)

3.3.4 Branching & Merging

VCS allows to make branches from the project and develop project separated from the original files. This functionality is often useful in software developing. When necessary changes are made in branched files, it is possible to merge changes back to original files. (Gyerik 2013, 10-12)

3.3.5 Initialize

Initialize is used to create a new, empty repository. This is always issued when a new project is started and it dows not share the common base with other software.

3.3.6 Checkout

Checkout is used to create a local working copy from the repository. If no additional parameters are given, checkout will obtain the latest revision of the software.

3.3.7 Commit

Commit is used to write the changes made in the working copy back to the repository. Committing will result in a new revision with commit message. Commit messages are used to help other developers to understand the changes.

3.3.8 Update

Update is used to merge the changes made in the repository into the local working copy.

3.3.9 Tag

A tag or a label is used to refer to an important snapshot in time for example release version of software.

3.3.10 Clone

Clone is used to create a copy of another repository containing all the revisions.

3.4 Version control workflow

3.4.1 Centralized workflow

The centralized workflow is used in the same way as working with CVCS's such as Subversion and it is the most often used workflow within the environments using CVCS. Users make changes and commit them to one central repository which acts as single point-of-entry for all changes. The workflow does not need any other

repositories than the one central repository. In Subversion this would be called as *trunk*.

Giving example with Subversion of very basic usage.

```
$ svn checkout

(time passes, changes are done)

$ svn commit -m ”New function xyzzy”
```

First developer clones the central repository using checkout. The developer makes changes to own local copy of the source code. After editing they commit changes using commit. If ”-m” option is not given, the commit command is followed with commit message input which is edited in text editor. If there are several developers working with the same codebase at the same time, developer might need to resolve conflicts. This happens if local changes diverge from the central repository. If conflict happens the developer have several options to choose from and the final output depends on how the developer want to resolve the conflict. (Centralized workflow)

An example with Subversion conflict as follows.

```
$ svn status
M       baz.c

$ svn commit -m "Update function xyzzy"
Sending        baz.c
Transmitting file data .
svn: E155011: Commit failed (details follow):
svn: E155011: File '/home/ojari/tmp/test/baz.c' is out of
date
svn: E170004: Item '/baz.c' is out of date

$ svn update
Updating '.'
C    baz.c
Updated to revision 3.
Summary of conflicts:
  Text conflicts: 1
Conflict discovered in file 'baz.c'.
Select: (p) postpone, (df) diff-full, (e) edit
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: dc
```

```
<<<<<<< MINE (select with 'mc') (1)
void xyzzy(float foo){}
||||||| ORIGINAL (1)
void xyzzy(int foo){}
=======
void xyzzy(double foo){}
>>>>>>> THEIRS (select with 'tc') (1)

Select: (p) postpone, (df) diff-full, (e) edit
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: tc
Resolved conflicted state of 'baz.c'
Summary of conflicts:
  Text conflicts: 0 remaining (and 1 already resolved)
```

Explaining the example above. First the developer tries to commit the changes assuming there is no changes in central repository. However, changes have occurred and there is a conflict which need to be checked first. The developer runs an SVN update command to get the latest changes from the central repository. The developer discovers one conflict in ”baz.c” file. The developer checks at least the conflicting parts of source code before making any decisions. In this case, the developer decided to use modification made by someone else as it fits better in this case.

3.4.2 Feature workflow

The feature workflow uses branches. Each time the developer starts working with particular feature, it uses dedicated branch. Dedicated branch helps users to work without disturbing the main codebase. Benefit of this workflow is that the main codebase should never be broken and the branches are focused to particular features and fixes. This workflow can be used with centralized or distributed VCSs.

Since the main code base is not used straigth forwardly, it is possible to use pull requests. Pull request is built-in Git functionality which generates summary of changes. For example, when the feature is finished, the developer makes a pull request. The project manager may decide to merge the changes to main codebase after the changes are reviewed and accepted by other developers. This functionality

can be used as a concept in other VCSs also, however, not as effectively as in Git. (Feature Branch workflow)

Giving example with Git of feature workflow with pull requesting.

```
$ git checkout work

(Making changes)

$ git commit
$ git tag -s -m "Completed xyzzy feature" feature-xyzzy work
$ git push https://example.com/foo.git/ +feature-xyzzy
$ git request-pull v1.0 https://example.com/foo.git/ feature-xyzzy >
message.txt

----- message.txt -----

The following changes since commit 203da7c294179...:
  Foo 1.0 (2016-12-03 10:12:05 +0200)
are available in the git repository at:
  https://example.com/foo.git tags/feature-xyzzy
for you to fetch changes up to 7a25a49c5835c...:
  Add return statement for xyzzy (2016-12-03 11:43:19 +0200)
----------------------------------------------
Completed xyzzy feature
----------------------------------------------

----- message.txt -----

(Review of pull request)

$ git checkout master
$ git pull https://example.com/foo.git/ tags/feature-xyzzy
```

Explaining the example above: the developer has set up the Git repository on the computer. The developer checkout the latest changes and starts working with the feature ”xyzzy”. The developer commits the changes and tags the commit with signature. The developer pushes the ”feature xyzzy” branch to the publishing repository. When the repository is pushed, the developer makes the message to request pull. The integrator reviews the pull request and decides to integrate the tag

in pull request. The integrator automatically makes merge commit as the integrator pulled a signed tag.

Practices for how to work may differ. Giving example how feature workflow could be done in Subversion using ”rebasing” method.

```
$ svn cp trunk feature

(time passes, new commits to feature & trunk)

$ svn cp trunk feature-rebase
$ svn co feature-rebase
$ cd feature-rebase
$ svn merge feature
$ svn commit
$ svn rm feature
$ svn mv feature-rebase feature
$ svn switch feature
$ svn merge --reintegrate feature
```

Explaining the example above. In this case the developer use feature branches. The developer starts to work with new feature. First the developer copies the *trunk* to new branch called *feature*. As the time goes on and the feature is ready, developer copies *trunk* to new branch, called *feature-rebase*. The developer changes the directory to *feature-rebase* and merges *feature* branch changes to *feature-rebase* branch. The developer commits the changes to central repository, removes the *feature* branch, renames the *feature-rebase* branch as *feature* and updates working copy URL. Now the developer is ready to issue merge action with ”reintegrate” option to replicate *feature* branch changes back into the *trunk*. The ”reintegrate” option is critical for reintegrating changes from a branch back into its original line of development.

3.4.3 Gitflow workflow

Gitflow workflow do not add new commands or concepts to what is required in centralized or feature workflow. However, it gives very specific meaning for different branches and how them are used. It bases on feature workflow with addition of using branches for development, releases and hotfixes. (Gtiflow workflow)

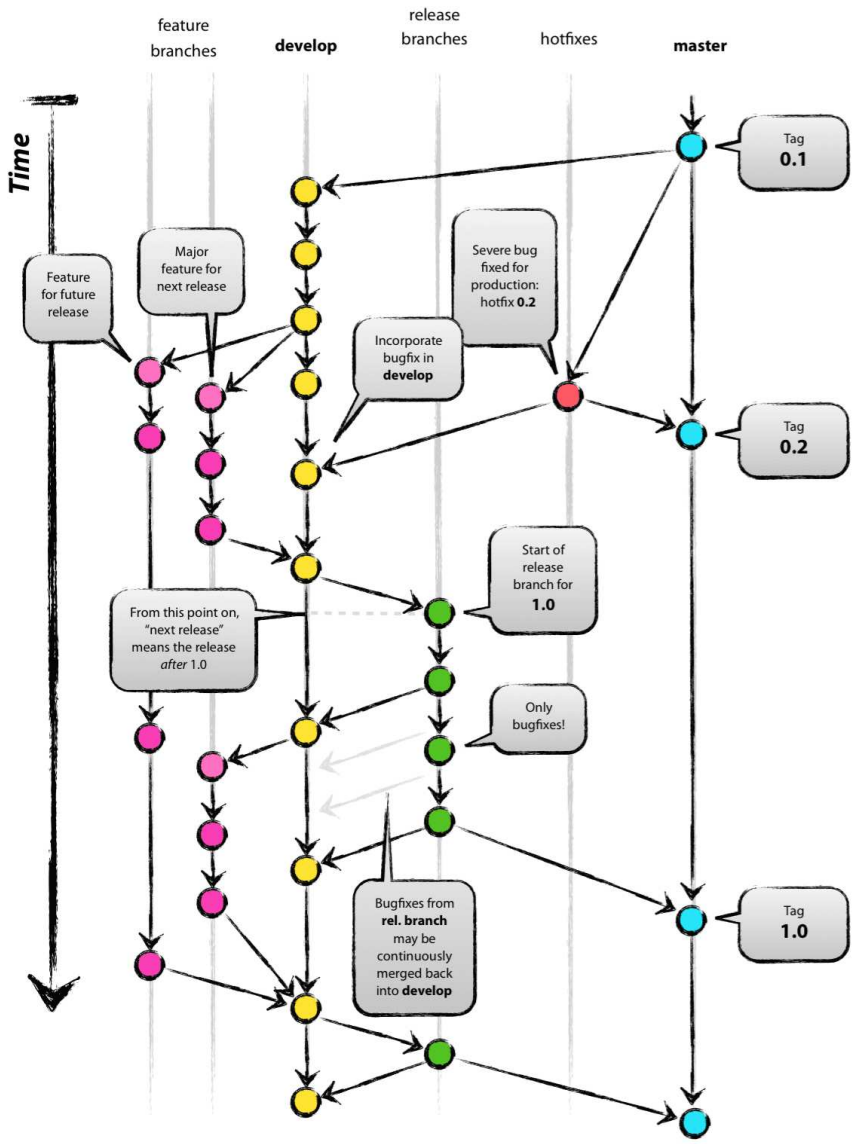

Figure 5. A Git branching model by Vincent Driessen (Source adapted from A successful branching Git branching model)

As illustrated in figure 5, starting with the main codebase, called as ”master” when using Git as VCS. Master branch is used to keep tagged stable releases. Master branch will never be broken and offers the latest stable release of the source code to others. Development branch is the place where ”bleeding-edge” version of the source code is. It is used for the continuous development. Feature branches are used by developers to make bigger changes to source code as described in the section before, but using the development branch as a main codebase when merging changes. Hotfixes are branches used to fix bugs. Release branches are bounded branches which will get only hotfixes, but no new features. Release branches are tagged and cloned to master branch which other users and developers sees first when they take a look at the repository.

3.5 IT service management

> *IT service management refers to the implementation and management of quality IT services that meet the needs of the business. IT service management is performed by IT service providers through an appropriate mix of people, process and information technology. (Bon 2009a, 340)*

The purpose of the IT Service Management (ITSM) is to make improvements which affects the organization directly or indirectly and increase the value of the company through benefits. Some generic benefits which are achieved through ITSM can be grouped into financial, employee, innovation and internal benefits. Within ITSM there will be costs involved, however, these costs must be compared with the cost of not implementing suitable processes. Mostly the organization will encounter issues with planning, implementing and running the ITSM processes. These issues can be easily overlooked, however, more often raised issues impact the realization of perceived benefits. (OGC 2006a, 9-13)

3.6 ITIL

> *A set of Best Practice guidance for IT Service Management. ITIL is owned by the OGC and consists of a series of publications giving guidance on the provision of Quality IT Services, and on the Processes and facilities needed to support them. (Bon 2009a, 340)*

ITIL was developed in recognition of the fact that organizations are becoming dependent on IT to meet strategic goals. IT service has to be reliable, consistent, of a high quality and of acceptable cost. ITIL was developed to disseminate proven ITSM best practices with a systematic approach to the delivery of quality IT services and developing processes which are effective and efficient. ITIL describes the relationships between the activities in processes and operations to attain the required quality. Processes together provide effective framework independently from the structure of the organization. (OGC 2006b, 19-21)

Besides the systematic approach through processes, ITIL offers the approach and philosophy shared by the people who work with it in practice through Service Lifecycle as the main structure for its guidance (Bon 2009b, 19-21).

3.7 ITIL v3

ITIL v3 consists of five phases in lifecycle publications, each volume providing guidance and detailed information to attain processes required by the ISO/IEC 20000 standard specification. The five phases are Service Strategy, Service Design, Service Transition, Service Operation and Continual Service Improvement. (OGC 2011, 5) The phases are illustrated in Figure 6.

Figure 6. The Service Lifecycle (Source adapted from Bon, J. von. 2009a)

3.7.1 Service Design

> *The Service Design stage takes business requirements and creates services, their supporting practices and management tools which meet business demands for quality, reliability, and flexibility (OGC 2008, 38)*

Service Design turns Service Strategy into a plan for delivering the business objectives. Service Design provides guidance for the design and development services and management practices. (OGC 2011, 6)

3.7.2 The five design aspects

An overall, integrated approach should cover the design of

- Service solutions, including all of the functional requirements, resources and capabilities needed and agreed
- Service Management systems and tools, especially the Service Portfolio for the management and control of services through their lifecycle
- Technology architectures and management architectures and tools required to provide the services
- Processes needed to design, transition, operate and improve the services
- Measurement systems, methods and metrics for the services, the architectures and their constituent components and the processes.

When a new or changed service solution is produced, it needs to be checked against each of the other aspects to ensure integration and interface with the existing services. (OGC Office of Government Commerce 2011, 30)

3.7.3 Designing service solutions

Formal and structured approach is needed to produce services within the right timeframe, functionality and cost. The approach must be iterative to ensure the services are aligned with business strategic goals, policies, infrastructure and requirements during the process development. (OGC Office of Government Commerce 2011, 51) An example of such an approach and its stages is illustrated in Figure 7.

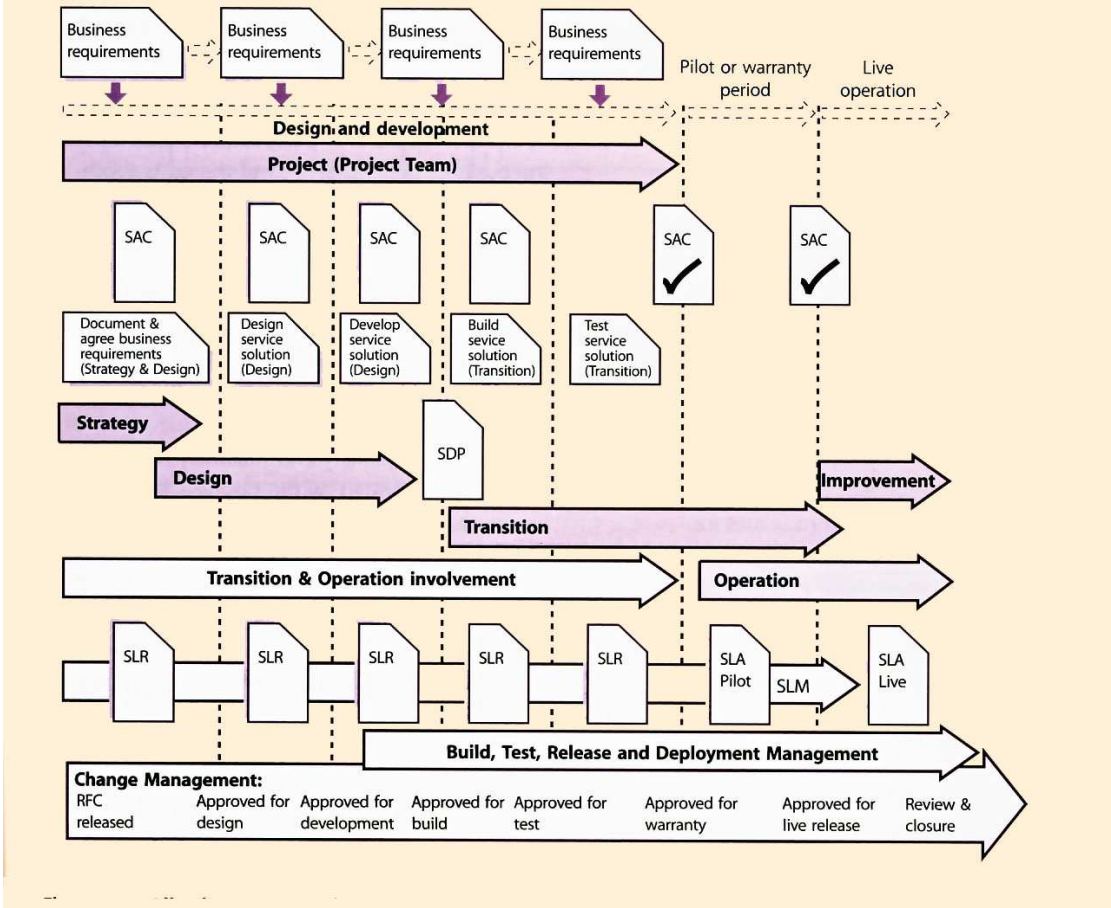

Figure 7. Aligning new services to business requirements (Source adapted from OGC Office of Government Commerce. 2011)

3.7.4 Designing management information systems and tools

The most effective way of managing services is using an appropriate management system. The most critical system from a Service Design point of view is service portfolio since it is used to support all processes. The service portfolio includes important components and ideally should form a part of the service knowledge management system (SKMS) and be registered as a document in configuration management system (CMS). (OGC Office of Government Commerce 2011, 52) The components of the service portfolio are illustrated in Figure 8.

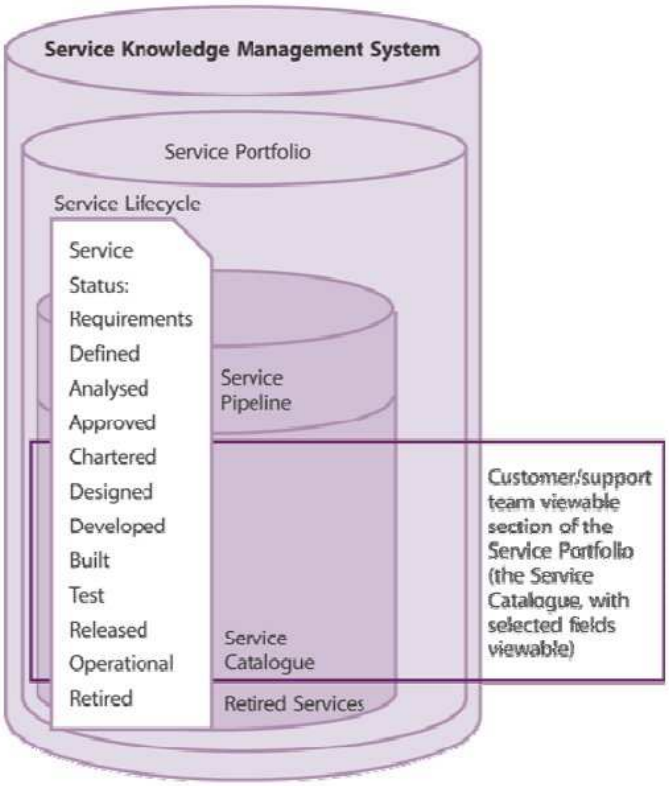

Figure 8. The service portfolio and its contents (Source adapted from OGC Office of Government Commerce 2011)

3.8 Virtualization

Virtualization is a technique which allows to run more than one operating system at a time. It enables to run software written for one operating system on another without rebooting between the operating systems. Virtualization allows easier software installations, testing and disaster recovery and it can reduce hardware and electricity costs by infrastructure consolidation. (Oracle VM VirtualBox User Manual, 12)

3.8.1 Virtual Machine

VM is the special environment that hypervisor creates for a guest operating system while it is running. In other words, the guest operating system is run “in” a VM. Normally, a VM will be shown as a window on a computer’s desktop. In a more abstract way, a VM is a set of parameters that determine its behavior. They include hardware settings and state information. (Oracle VM VirtualBox User Manual, 13)

3.8.2 Hypervisor

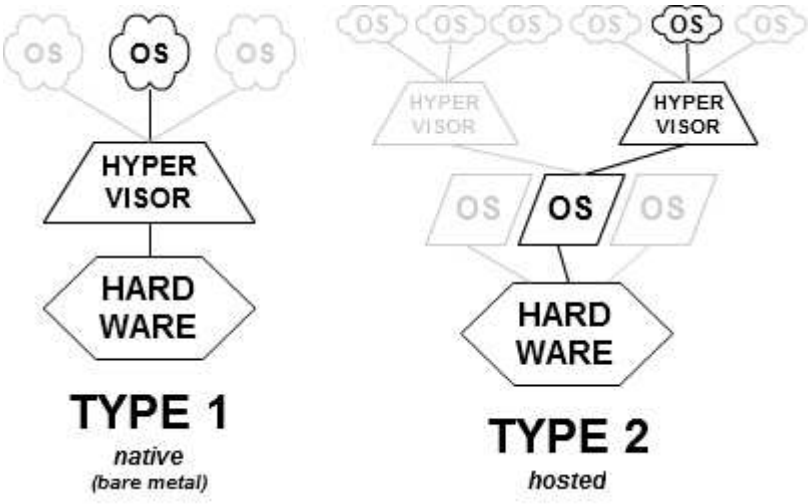

Figure 9. The two types of Hypervisors (Source adapted from Hypervisor – Wikipedia)

A hypervisor is a hardware virtualization technique that allows multiple guest operating systems to run on a single host system at the same time. The guest operating system shares the hardware resources of the host computer, such that each guest appears to have its own hardware resources. Hypervisors can be divided into two types as shown in Figure 9.

Type 1 hypervisors, also known as native or bare-metal hypervisors, runs directly on the host computer's hardware to control the hardware resources and to manage guest operating systems.

Type 2 hypervisors, also known as hosted hypervisors, runs within a formal operating system environment. This type of hypervisor runs as a distinct second layer while the operating system runs as a third layer above the hardware. (What is Hypervisor - Definition from Techopedia)

3.9 Data storing and backup

3.9.1 Network Attached Storage

Network Attached Storage (NAS) is a storage device connected to a network. Using a NAS server allows to share files among other clients in the network. The clients connect to the NAS through a computer network which allows the NAS to be located

anywhere in the network. The NAS device may include the server system within the device and are often referred as NAS servers. Depending of the NAS server, it may contain several hard disk drives with RAID capability. (What is NAS? - Buffalo Technology)

3.9.2 RAID technologies

Redundant Array of Independent Disks (RAID) is data storage technology that combines multiple hard disk drives into an array and can be used like a one hard disk drive. There are variety of RAID modes which offer different levels of integrity and fault tolerance. (What is RAID? - Buffalo Technology)

Mirrored (RAID 1)

In mirroring the data is written to two disks, so that there are always two copies of the same information. Data can be retrieved from the disk with the shorter delays and if a disk fails, the second copy can be used. Mirroring is used in applications where availability and transaction rate are more important than storage efficiency (Chen 1994, 10)

3.9.3 Securing data with backup

A backup is a secondary copy of data used for data protection. Backing up data should not be confused with archiving data, where the primary data is moved to a less-expensive type of media such as tape for long-term and low-cost storage. The purpose of the backup is to create a copy of data so it may be restored after data is lost, corrupted, deleted or if a disaster strikes.

The objective of backup is to ensure data retrieval in any situation if the data is lost in the primary location. Backup and recovery testing examines practices and technologies for data security and data replication. With periodic testing, it is guaranteed that the goal of protecting data is being met. (Full, incremental or differential 2008)

Full backup

Full backup is the most basic and complete type of backup operation and it forms a base for other backups. The primary advantage is that Full backup makes a copy of all

data to a single set of media. This results in a minimal time to restore data from other types, however, the disadvantages are that it takes longer to perform and it requires more storage space. (Full, incremental or differential 2008)

Incremental backup

An incremental backup will result in copying only the data that has changed since the last backup operation of any type. Incremental backup may be run almost as often as desired because it will copy only the most recent changes stored. Incremental backup results in a smaller amount of data than a full backup. The advantage of incremental backup is that it will complete faster, and require less media to store the backup. The disadvantage is that in the case of data restore the entire initial backup and each of the previous backups are needed to retrieve all the updated files. (Full, incremental or differential 2008)

Differential backup

Differential backups sit between the Full and incremental backup. A differential backup creates an independent file, containing all changes since the last full backup. Differential backup does not have to process through all previous backups in case of data restore which makes it faster than incremental backup. Differential backup requires more space and time to complete than incremental backups, however, less than a full backup. (Full, incremental or differential 2008)

3.9.4 Bacula backup system

Bacula is an enterprise level computer backup system, designed to automate backup tasks in a heterogeneous networks. Bacula is a very comprehensive set of software used to manage backup, recovery and verification of computer data across a network of computers. Bacula is a network based backup program and it is capable of handling systems consisting of hundreds of computers located over a network. Bacula can run on a single computer or it can be distributed depending on the configuration. As illustrated in Figure 10, Bacula is made up of five major components which are: Director, Console, File, Storage, and Monitor services. (Sibbald 2016, 1-2)

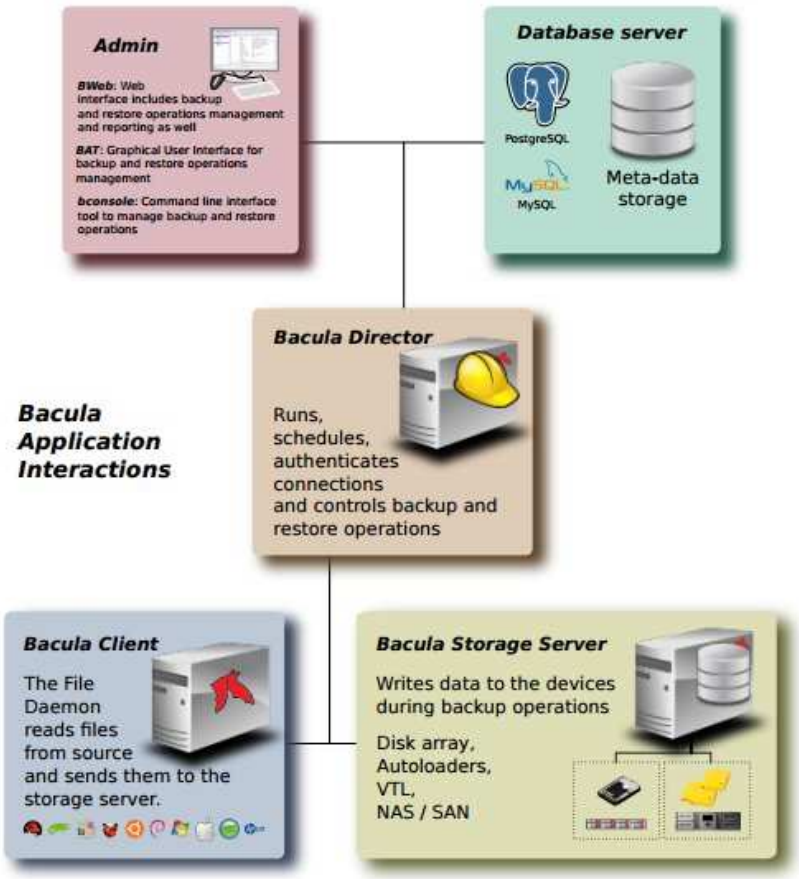

Figure 10. Bacula components (Source adapted from Sibbald 2016)

Typical interactions between the Bacula Services for a backup job are presented in Figure 11 where each block represents a seperate process in general. Each service is presented in the following chapters. (Sibbald 2016, 7-8)

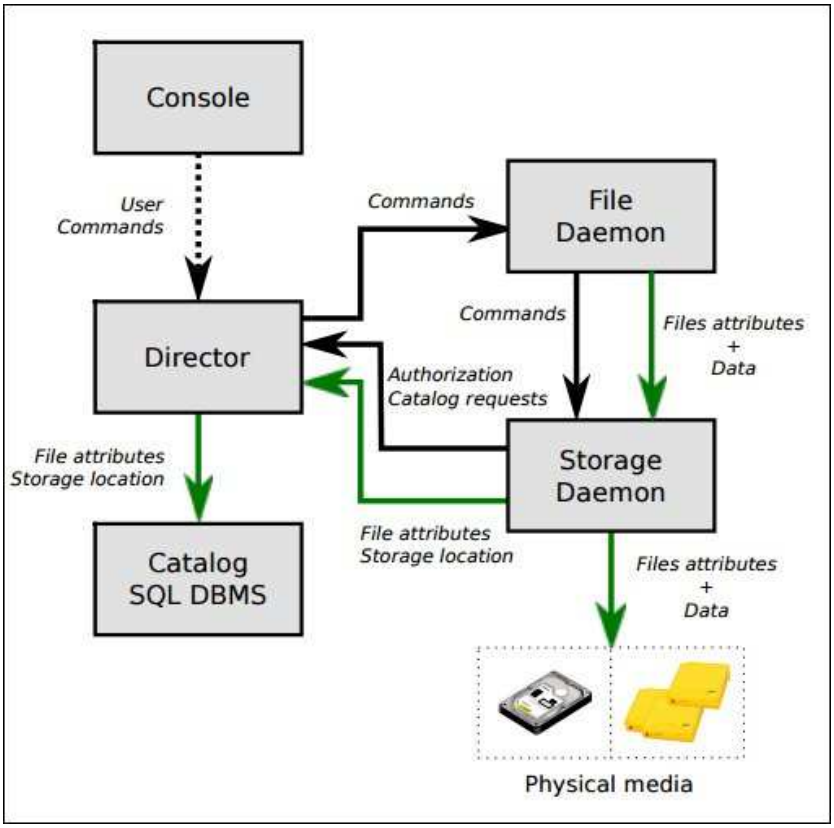

Figure 11. Interactions Between the Bacula Services (Source adapted from Sibbald 2016)

Bacula Director

The Bacula Director service supervises all the backup, restore, verify and archive operations. The system administrator uses the Bacula Director to schedule backups and to recover files. (Sibbald 2016, 2)

Bacula Console

The Bacula Console service allows the administrator to communicate with the Bacula Director. The Bacula Console is available in three versions: text-based console interface, QT-based interface, and a wxWidgets graphical interface. In this case, text-based console interface is used. (Sibbald 2016, 2)

Bacula File

The Bacula File service is installed on the machine to be backed up. It is specific to the operating system on which it runs and is responsible for providing the file attributes and data when requested by the Director. The File services are also responsible for the file system dependent part of restoring the file attributes and data during a recovery operation. (Sibbald 2016, 2-3)

Bacula Storage

The Bacula Storage services consist of the software programs that perform the storage and recovery of the file attributes and data to the physical backup media or volumes. The Storage daemon is responsible for reading and writing files. (Sibbald 2016, 3)

Bacula Catalog

The Catalog services are comprised of the software programs responsible for maintaining the file indexes and volume databases for all files backed up. The Catalog services permit the system administrator to quickly locate and restore any desired file. The catalog maintains a record of all Volumes used, all Jobs run, and all Files saved, permitting restoration and Volume management. Bacula currently supports three different databases, MySQL, PostgreSQL, and SQLite, one of which must be chosen when building Bacula (Sibbald 2016, 3)

Bacula Monitor

A Bacula Monitor service is the program that allows the administrator to watch current status of Bacula Directors, Bacula File Daemons and Bacula Storage Daemons. To perform a successful save or restore, the following four daemons must be configured and running: the Director daemon, the File daemon, the Storage daemon, and the Catalog service with MySQL, PostgreSQL or SQLite database. (Sibbald 2016, 3)

4 System design

4.1 Best practices

To achieve a well defined IT system, the processes presented in ITIL were studied and used when possible. As the ITIL framework is very comprehensive publication series and a huge process to fully cover, only the necessary sections were covered. The workload was reduced by only covering the Service Design publication and related literature as a reference. ITIL Service Design publication covers all the needed procedures, tasks, and checklists to achieve better design of the IT system during

design phase. Despite the study of the ITIL, the focus is in requirement specification and ITIL was used as reference to adapt paradigm of the design process.

4.2 Design overview

Overview of the network structure is shown in Figure 12. All the devices are connected through one central switch which forms a star network topology. The switch can be used to control and inspect data traffic in the network for administration purposes.

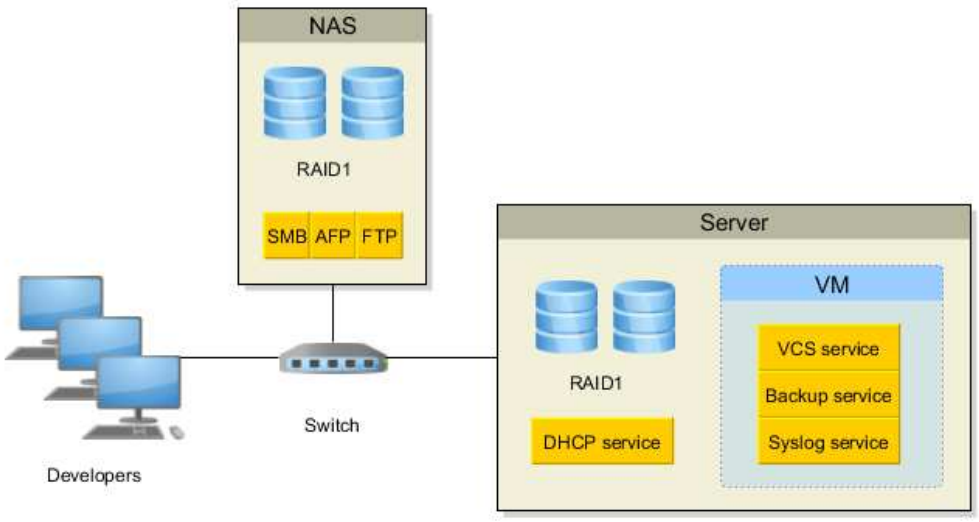

Figure 12. The IT system design LAN topology

When a device is connected to a network, it obtains the IP address from DHCP service installed in the server. The switch, NAS and one VM have dedicated static IP addresses and other computers obtain the first free IP address in order. The release times of the IP addresses are set to one year to make working with different devices smoother. The IP address pool should not fill-up as the devices connected to the network are limited.

The server has a basic setup and configuration to run VMs. VMs can be built and tested somewhere else before placing them to the server which makes further development easier in the future. One VM is installed and configured to run VCS, system logging and backup services.

Backups are implemented to NAS which is an independent system and physically separate from the server. NAS could be located in a different place than the server in case of a disaster. In this case the NAS is first located in the same building as the server and may be moved to another location when the system is configured and fully tested.

The integrity of the system and NAS is maintained by using RAID1 hard drive setup and scheduled file system checks. Using the RAID1 setup, all the data is copied at least on two hard drives which decreases the possible data loss during hardware failures. File system checks maintain the file system integrity and are done on a a regular basis or when a sudden power off has occurred in the system.

To keep a record on how the system is performing, all actions are recorded to log files. The devices supporting a logging and external logging server are configured to send the log files to the server. In the case of a programmatic error, notification is sent via SMTP service to the local administration account in a server used for monitoring of the services.

The network is isolated from other networks, especially from the internet. The network needs to be hardened and fully tested to make sure it is secure from outer threats. There was also no reason to connect to the system to the internet. All the necessary installation of the server base system is done before it is isolated from other networks. Afterwards the server can be updated with removable media such as CD or USB memory. VMs can be reconfigured and updated in other computers and then moved to the server.

4.3 Hardware

The IT System setup consists of one PC, NAS and switch. The PC acts as a server and a base for all the services, NAS is a separate physical place for data backups and switch connects all the devices and computers together to form the LAN.

The server was built using already available parts. For a better performance, the central memory units were replaced with bigger units, and the old hard drive was replaced with two identical bigger hard drives. All other necessary equipment such as

keyboard and display were found in the company. The final technical specifications are illustrated in Table 1.

Table 1. Server technical specifications

UNIT	Acer Veriton M265
CPU	Intel® Pentium® Processor E5300 (2M Cache, 2.60 GHz, 800 MHz FSB)
RAM	1: Kingston 2GB DDR2 800MHz CL6 (KVR800D2N6/2G) 2: Kingston 2GB DDR2 800MHz CL6 (KVR800D2N6/2G)
STORAGE	HDD1: Seagate Barracuda 1TB, 64MB 7200 RPM 3.5" SATA III, 6 Gb/s (ST1000DM003) HDD2: Seagate Barracuda 1TB, 64MB 7200 RPM 3.5" SATA III, 6 Gb/s (ST1000DM003)

The chosen NAS was Buffalo Linkstation™ 441 (LS441) which is designed for business and home office use. It is a ready standalone storage solution with internal server equipment. LS441 is able to perform automated integrity tests which is very important feature in the setup. LS441 has four hard drive places with RAID 1/5/10 capability and gigabit network peripheral which is enough for the current usage.

The chosen switch was HP1910 Ethernet switch which included 24 connections and is capable for layer 3 networking. This switch was chosen because it has wide range of configuration options to support the administration and future additions.

4.4 The Base system

The operating system had to be lightweight to save system resources. The chosen operating system for the server was Linux Ubuntu Server as the other software developers in the company are familiar with the Ubuntu environment. It is also a convenient choice because it is open source software with a wide community support (Canonical Group Ltd 2016a) and it is licensed free for commercial use (Canonical Group Ltd 2016b).

The operating system partition scheme for the server haas “boot”, “root”, “srv” and “swap” partitions. The “boot” partition is separated for better fault tolerance as it

allows the system to boot as long the hardware is working. The “root” partition was kept relatively small to keep the automatic file system checks short. The “swap” partition is implemented to cover cases when the memory is running out. This situation should never happen, however, the ”swap” partition may rescue the system from complete system halt if there is a problem which consumes plenty of memory in a server. The “srv” partition will work as a primary partition for services provided by the server for example the VMs are installed in “srv” partition. All partition are configured to RAID 1 volume except the boot partition, which allows legacy support for older GRUB and BIOS systems, making the recovery of the system easier in case of hardware or software failures.

SSH, virtualization, and DHCP software were installed on top of the operating system installation. SSH is used to get access to the system from other computers in LAN if maintenance is necessary. Virtualization is used to run VMs in the server. The chosen hypervisor for virtualization was “Oracle VM VirtualBox”, a type 2 hypervisor. It was chosen because the other developers are familiar with the software. It is also licensed with GNU GPLv2, which means it can be used in the company with some limitations (Licensing_FAQ - Oracle VM VirtualBox).

The graphical user interface (GUI) was decided to be installed to allow control and monitoring locally. GUI is not recommended in server environments because it uses resources (ServerGUI - Community Help Wiki). To avoid bloat software and keep GUI relatively light, it was installed as separate packages. For having GUI in Linux environment, there are three main components to be installed: Display server, Display manager and window manager. The operating system installation was already included with a X Window System as a display server. The chosen display manager was “lightdm” and window manager was “i3” tiling window manager. The desktop is always started as an empty screen with the status bar at the bottom, however, it still provides a fully functional lightweight desktop environment to launch applications with GUI. An example of i3 tiling layout is shown in Figure 13.

Figure 13. I3 container layout example (Source adapted from Stapelberg 2013)

4.5 Configuring VM

One VM is installed with the same operating system as the base system and the services are placed there. This chapter provides a short overview on the services installed and configured to the VM.

TFTP

The TFTP server was installed with the following command

```
sudo apt-get install tftpd-hpa
```

Once the installation is complete, TFTP server is running on the system and will be listening on all active network interfaces, on both IPv4 and IPv6. By default, TFTP now allows other devices to download files from the TFTP server.

NFS

The NFS server was installed with the following command

```
sudo apt-get install nfs-kernel-server
```

The configuration was implemented by adding the directories to be exported to /etc/exports file. For example

```
/ubuntu   *(ro,sync,no_root_squash)
/home     *(rw,sync,no_root_squash)
```

One can replace * with one of the hostname formats to restrict the access to the NFS mount.

To start the NFS server, the following command is run at a terminal prompt

```
sudo service nfs-kernel-server start
```

Samba

The Samba server was installed with the following command

```
sudo apt-get install samba
```

The main Samba configuration file is located in /etc/samba/smb.conf. The configuration was implemented by adding an own section to the /etc/exports file which describes the share folder. For example

```
[share]
    comment = Ubuntu File Server Share
    path = /srv/samba/share
    browsable = yes
    guest ok = yes
    read only = no
    create mask = 0755
```

Finally, the Samba server services are restarted with the following commands

```
sudo restart smbd
sudo restart nmbd
```

SVN

The SVN server was installed with the following command

```
sudo apt install subversion
```

After the installation, the Subversion repository was created. For example

```
svnadmin create /path/to/repos/project
```

Once the repository is created, files are imported into the repository. For example

```
svn import /path/to/import/directory \
file:///path/to/repos/project
```

Apache Web Server

The Apache Web Server was installed with the following command

```
sudo apt-get install apache2
```

By default, content in /var/www/ is shown over HTTP protocol and can be accessed from LAN.

Git Server

The Git server was installed with the following command

```
sudo apt-get install git-core
```

A new repository is created with the following commands

```
mkdir test-repo.git
cd test-repo.git
git --bare init
```

To make the Git Server work over HTTP protocol, a repository was created under /var/www/ folder and permissions were changed. For example

```
cd /var/www
mkdir test-repo.git
cd test-repo.git
git --bare init
git update-server-info
chown -R www-data:www-data
```

WebDAV plugin on Apache Web Server needed to be enabled with the following command

```
sudo a2enmod dav_fs
```

Then /etc/apache2/mods-enabled/dav_fs.conf was filled with a new section. For example

```
<Location /test-repo.git>
        DAV on
        AuthType Basic
        AuthName "Git"
        AuthUserFile /etc/apache2/passwd.git
        Require valid-user
</Location>
```

The user account needed to be added to allow the access to repository with the following command

```
htpasswd -c /etc/apache2/passwd.git testUser
```

Now the user is prompted to enter the password for “testUser”. These credentials are used to communicate with Git. To apply changes, Apache Web Server is to be restarted with the following command

```
/etc/init.d/apache2 restart
```

Now the client can access the repository. For example

```
mkdir ~/Git/test-project
cd ~/Git/test-project
git init
git remote add origin \
http://testUser@example.com/test-project.git
touch README.md
git add .
git commit -a -m “Initial import”
git push origin master
```

4.6 Why Bacula

The chosen software for backup and recovery was Bacula backup solution. The backup solution was chosen due to the recommendations from different communities and websites which pointed to Bacula. The recommendations were mostly justified based on the configurability and stability of the software. While exploring Bacula system there was much discussion about BareOS and BURP backup systems, which led to a post by Kern Sibbald (Sibbald 2014) where it was stated that these were forks from the original Bacula project. After exploring these solutions, the conclusion was made to use Bacula software. The reason for this was that Bacula may be more mature than the other derived projects and the documentation is very comprehensive, which is helpful in many situations.

4.6.1 Configuring Bacula

In this thesis work, all the components are installed and configured on single VM. Bacula configuration mainly consists of four configuration files as illustrated in Figure 14.

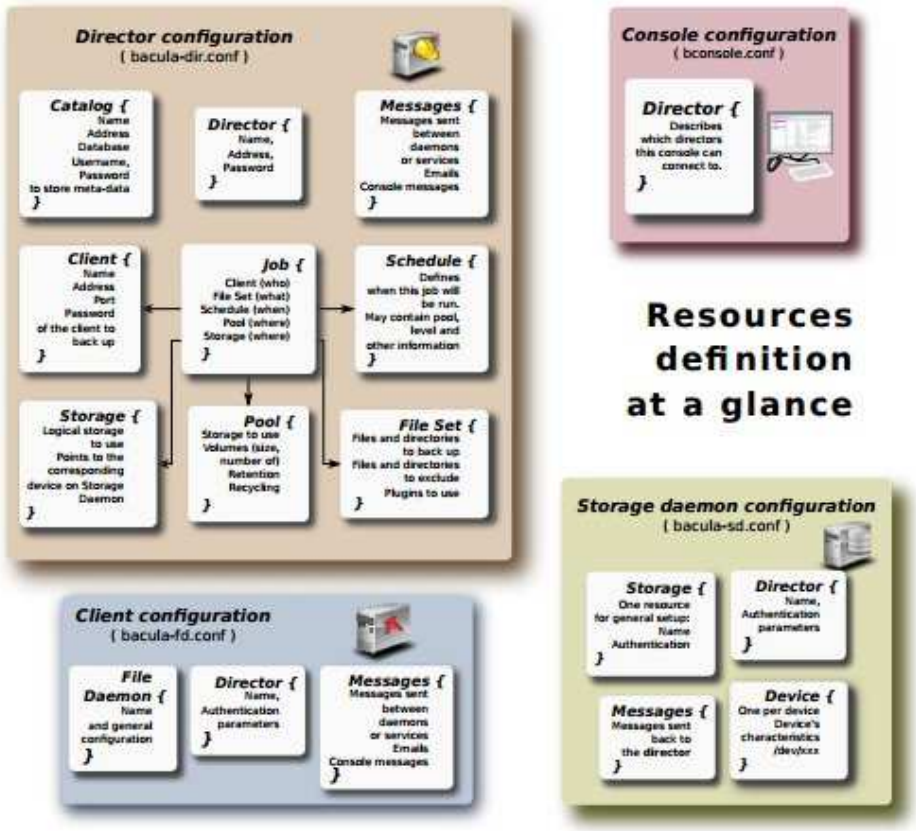

Figure 14. Bacula resources definition (Source adapted from Sibbald 2016)

The configuration of Bacula was fairly easy as it is mostly pre-configured. Only a small portion of configuration was needed to get the Bacula services to communicate with each other. The backup plan was configured according to Bacula Main Reference overall design example (Sibbald 2016, 261 - 266). With the configuration, Bacula maintains six months of backup data and is able to access the old files on a daily basis for a week, a weekly basis for a month, then monthly for six months. The full backup is done once a month, a Differential backup once a week, and Incremental backup daily. The detailed example configuration files can be found in Appendices 1, 2, 3 and 4.

4.7 Security

According to the requirement specification, the system is not connected to WAN, therefore security breaches through the internet should not be possible. There is a physical access to the system, however, the access to the company's office is

controlled, which why the security is not the main concern. In sense of good practice, the basic security is implemented to prevent a prohibited access in the LAN by using credential authorization for the devices and services.

5 Testing

REQ01: Server MUST automatically backup files to Network Attached Storage (NAS) - frequently changing files once a day and full system backup once a month.
REQ02: Every successful backup MUST be logged.
REQ03: Server MUST maintain the integrity of the files.
REQ04: Server files MUST be mirrored at least on two hard drives.
REQ05: Server and services MUST be reached at normal working time in minimal delay.
REQ06: Server configuration MUST be moveable to another system.
REQ07: Server files MUST be able to recover after system failure.
REQ08: Server MUST be able to recover from programmatic errors after user input.
REQ09: Server MUST be able to detect and inform the user of possible failures.
REQ10: Server network MUST be isolated from other networks.

The system was tested according to the requirements specification. To fulfill REQ01, an automatic backup was confirmed to work by checking the backup files existing in NAS after few backups. To fulfill REQ02, it was checked that the backups are logged by the Bacula software which sends message through SMTP to the local user. To fulfill REQ03, integrity of the NAS and the server was confirmed to be working by checking the procedures after the forceful power off. To fulfill REQ04, RAID functionality of the server and the NAS was confirmed to work by disconnecting one of the hard drives without powering off the system. To fulfill REQ05, all the services were confirmed to be reachable without external delays by using the services during normal work time. To fulfill REQ06, portability of the configuration was tested by starting the VM in different computer successfully. To fulfill REQ07, the recovery of the files was tested with a full backup and confirmed to be working by extracting the backup files from NAS to server. To fulfill REQ08, it was tested that the system was able to recover from

programmatic errors after user input during heavy configuration. To fulfill REQ09, the failures caused by the software were informed to a local user account through SMTP by forcefully closing a running service. To fulfill REQ10, the system LAN was confirmed to be isolated from WAN by using network mapper “nmap” scanning tool in the LAN.

6 Evaluation

The server and the host system haves been online for one year after the implementation. The server and network infrastructure design was successful and meets the requirements set by the specification. The implementation process went well. By using the VMs, it was possible to expand the system when needed and test different configurations without disconnecting or breaking currently running services. The availability of the services has been very high, and the system seems to be very fault tolerant in cases of power failures. Data backup service runs as expected, however, data recovery is only partially tested and it needs further testing to confirm data integrity in the future.

7 Follow-up

At the time of writing, the maintenance of the system is fairly complex as it consists of many different software components. The developers working currently are familiar with the new set up to make further configurations. To ensure, the system is usable and can be developed in the future, a more detailed documentation of the system and the services should be available.

An automatic testing of the IT system could improve the reliability of the system. This could be accomplished by regularly executing dedicated testing software and tailored hand written scripts. The automatic testing could include hardware testing, system software testing and data backup testing.

VMs use a great deal of system resources which may result in problems if there are many VMs. An alternative choice for VMs could be application containers which do not need the virtualization of the hardware.

8 Conclusion

The VCS server design was successful and the implementation process went well. The study of ITIL framework gave more understanding about ITSM and broadened my perspective for design work as I was more aware of how an IT system should be designed to support the business strategy. The use of ITIL framework helped to achieve a better design of the IT system during the design phase. ITIL is a very comprehensive framework which gives a well-structured basis for the IT system. However, the ITIL framework is a huge subject to study and a burdensome task to fully implement.

The use of VMs in the design allows to develop the server environment in stages without interfering with it. VMs can be tested before implementing them in the server which is a big benefit for quality assurance. The server design allows continuous development, which is necessary to ensure the server can be updated and new services can be implemented as necessary.

The objectives of the thesis were met and Senop Oy R&D team were pleased from the server set up. The server set up supports the software development work and it had a positive influence to the software development in the R&D team.

References

Bon, J. von. 2009a. Foundations of IT Service Management based on ITIL V3. Zaltbommel: Van Haren.

Bon, J. von. 2009b. IT Service Management Based on ITIL V3 - A Pocket Guide. Zaltbommel: Van Haren.

Chacon, S. & Straub, S. 2014. Pro Git. E-book. Accessed on 12.3.2016. Retrieved from https://progit2.s3.amazonaws.com/en/2016-03-08-3f34a/progit-en.1067.pdf

Canonical Group Ltd. Licensing. Accessed on 2.10.2016a. Retrieved from https://www.ubuntu.com/about/about-ubuntu/our-philosophy

Canonical Group Ltd. Licensing. Accessed on 2.10.2016b. Retrieved from https://www.ubuntu.com/about/about-ubuntu/licensing

Data backup types explained: Full, incremental, differential and incremental-forever backup. Referenced 20.10.2016. Retrieved from http://searchdatabackup.techtarget.com/tip/Data-backup-types-explained-Full-incremental-differential-and-incremental-forever-backup

Department of Energy Quality Managers. 2000. Software Configuration Management (SCM)

A Practical Guide. Accessed on 12.11.2016. Retrieved from http://energy.gov/sites/prod/files/cioprod/documents/scmguide.pdf

A successful Git branching model. Accessed on 4.12.2016. Retrieved from http://nvie.com/posts/a-successful-git-branching-model/

Centralized workflow. Accessed on 4.12.2016. Retrieved from https://www.atlassian.com/git/tutorials/comparing-workflows/centralized-workflow

Feature Branch workflow. Accessed on 4.12.2016. Retrieved from https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Full, incremental or differential: How to choose the correct backup type. 25.8.2008. Accessed on 20.10.2016. Retrieved from http://searchdatabackup.techtarget.com/feature/Full-incremental-or-differential-How-to-choose-the-correct-backup-type

Gitflow workflow. Accessed on 4.12.2016.
Retrieved from Retrieved from https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Gyerik, J. 2013. Bazaar version control : a fast-paced practical guide to version control using Bazaar. E-book. Accessed on 3.4.2016. Retrieved from https://jyu.finna.fi/Record/jykdok.1439749

Hypervisor - Wikipedia. Accessed on 25.10.2015 Retrieved from https://en.wikipedia.org/wiki/Hypervisor

Kananen, J. 2009. Toimintatutkimus yritysten kehittämisessä. Sarja: Jyväskylän ammattikorkeakoulun julkaisuja 101. Jyväskylä: Yliopistopaino.

Kananen, J. 2013. Design Research (Applied Action Research) as Thesis Research. Sarja: Publications of JAMK University of Applied Sciences 146. Tampere: Yliopistopaino.

Sibbald, K. 14.9.2014. Why forking is bad | Bacula. Accessed on 23.3.2016.

Retrieved from http://blog.bacula.org/why-forking-is-bad/

Licensing_FAQ - Oracle VM VirtualBox. Accessed on 15.10.2016. Retrieved from https://www.virtualbox.org/wiki/Licensing_FAQ

LTS - Ubuntu Wiki. Accessed on 15.10.2016. Retrieved from https://wiki.ubuntu.com/LTS

OGC Office of Government Commerce. 2011. ITIL Service Design. TSO The Stationery Office.

OGC Office of Government Commerce. 2006a. Planning to Implement Service Management. TSO The Stationery Office.

OGC Office of Government Commerce. 2006b. Introduction to ITIL. TSO The Stationery Office.

OGC Office of Government Commerce. 2008. ITIL V3 Foundation Handbook. TSO The Stationery Office.

Oracle VM VirtualBox User Manual. Referenced 29.10.2016

Retrieved from http://download.virtualbox.org/virtualbox/UserManual.pdf

Patria Oyj. 2016. Senop focuses on advanced sensor technology for defense and security. 29th January 2016. Accessed on 5.6.2016. Retrieved from http://patria.fi/en/media/news/senop-focuses-advanced-sensor-technology-defense-and-security-markets

Chen, P., Lee, E., Gibson, G., Randy, K. & Patterson, D. 1994. RAID: High-Performance, Reliable Secondary Storage. E-book. Accessed on 6.12.2016. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.3889

ServerGUI - Community Help Wiki. Referenced 15.10.2016. Retrieved from https://help.ubuntu.com/community/ServerGUI

Sibbald, K. 2016. Bacula Main Reference. 21.9.2016. Accessed on 25.10.2016. Retrieved from http://www.bacula.org/7.4.x-manuals/en/main/main.pdf

Stapelberg, M. 2013. i3 User's Guide. Accessed on 20.10.2016 Retrieved from https://i3wm.org/docs/userguide.html

Version Control by Example. A History of Version Control. Accessed on 3.12.2016. Retrieved from http://ericsink.com/vcbe/html/history_of_version_control.html

What is Apache Subversion (SVN)? - Definition from Techopedia. Accessed on 3.10.2016 Retrieved from https://www.techopedia.com/definition/3304/apache-subversion-svn

What is Apache Web Server - Definition from Techopedia. Accessed on 3.10.2016 Retrieved from https://www.techopedia.com/definition/4851/apache-web-server

What is DHCP? - Definition from Techopedia. Techopedia dictionary. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/11337/dynamic-host-configuration-protocol-dhcp

What is Git? - Definition from Techopedia. Accessed on 3.10.2016
Retrieved from https://www.techopedia.com/definition/28960/git

What is Hypervisor - Definition from Techopedia. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/4790/hypervisor

What is Information Technology Infrastructure Library (ITIL) - Definition from Techopedia Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/24417/information-technology-infrastructure-library-itil

What is NAS? - Buffalo Technology. Accessed on 7.12.2016.
Retrieved from http://www.buffalo-technology.com/en/technology/standard-technologies/what-is-nas/

What is a Network File System (NFS)? - Definition from Techopedia. Accessed on 3.10.2016 Retrieved from https://www.techopedia.com/definition/1845/network-file-system-nfs

What is Redundant Array of Independent Disks (RAID)? - Definition from Techopedia. Accessed on 3.10.2016 Retrieved from https://www.techopedia.com/definition/24492/redundant-array-of-independent-disks--raid

What is RAID – Buffalo Technology. Accessed on 6.12.2016.
Retrieved from http://www.buffalo-technology.com/en/technology/standard-technologies/what-is-raid/

What is Revision Control System (RCS)? - Definition from Techopedia. Techopedia dictionary. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/30666/revision-control-system-rcs

What is a Samba? - Definition from Techopedia. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/3527/samba

What is Server Message Block (SMB)? - Definition from Techopedia. Techopedia dictionary. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/5470/server-message-block-smb.

What is a Trivial File Transfer Protocol (TFTP)? - Definition from Techopedia. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/1881/trivial-file-transfer-protocol-tftp

What is a Virtual Machine (VM)? - Definition from Techopedia. Accessed on 3.10.2016. Retrieved from https://www.techopedia.com/definition/4805/virtual-machine-vm

Appendices

Appendix 1 Bacula Console configuration

```
#
# Bacula User Agent (or Console) Configuration File
#

Director {
  Name = bacula-dir
  DIRport = 9101
  address = localhost
  Password = " *** CHANGE ME ***"
}
```

Appendix 2 Bacula Client configuration - File Daemon

```
#
# Default  Bacula File Daemon Configuration file
#
#  For Bacula release 7.0.5 (28 July 2014) -- ubuntu 16.04
#
# There is not much to change here except perhaps the
# File daemon Name to
#

#
# List Directors who are permitted to contact this File daemon
#
Director {
  Name = bacula-dir
  Password = " *** CHANGE ME ***"
}

#
# Restricted Director, used by tray-monitor to get the
#   status of the file daemon
#
Director {
  Name = bacula-mon
  Password = " *** CHANGE ME ***"
  Monitor = yes
}

#
# "Global" File daemon configuration specifications
#
FileDaemon { # this is me
  Name = bacula-fd
  FDport = 9102 # where we listen for the director
 WorkingDirectory = /var/lib/bacula
  Pid Directory = /var/run/bacula
  Maximum Concurrent Jobs = 20
# Plugin Directory = /usr/lib/bacula
  FDAddress = 127.0.0.1
}

# Send all messages except skipped files back to Director
Messages {
  Name = Standard
  director = bacula-dir = all, !skipped, !restored
}
```

Appendix 3 Bacula Director configuration

```
Director { # define myself
Name = bacula-dir
DIRport = 9101
QueryFile = "/home/bacula/bin/query.sql"
WorkingDirectory = "/home/bacula/working"
PidDirectory = "/home/bacula/working"
Maximum Concurrent Jobs = 1
Password = " *** CHANGE ME ***"
Messages = Standard
}

# By default, this job will back up to disk in /tmp
Job {
Name = client
Type = Backup
Client = client-fd
FileSet = "Full Set"
Schedule = "WeeklyCycle"
Storage = File
Messages = Standard
Pool = Default
Full Backup Pool = Full-Pool
Incremental Backup Pool = Inc-Pool
Differential Backup Pool = Diff-Pool
Write Bootstrap = "/home/bacula/working/client.bsr"
Priority = 10
}

# Backup the catalog database (after the nightly save)Job {
Name = "BackupCatalog"
Type = Backup
Client = client-fd
FileSet="Catalog"
Schedule = "WeeklyCycleAfterBackup"
Storage = File
Messages = Standard
Pool = Default
# This creates an ASCII copy of the catalog
# WARNING!!! Passing the password via the command line is insecure.
# see comments in make_catalog_backup for details.
264 Bacula Version 7.4.4
RunBeforeJob = "/home/bacula/bin/make_catalog_backup bacula bacula"
# This deletes the copy of the catalog
RunAfterJob = "/home/bacula/bin/delete_catalog_backup"
Write Bootstrap = "/home/bacula/working/BackupCatalog.bsr"
Priority = 11 # run after main backup
}

# Standard Restore template, to be changed by Console program
Job {
Name = "RestoreFiles"
Type = Restore
Client = havana-fd
FileSet="Full Set"
Storage = File
Messages = Standard
```

```
Pool = Default
Where = /tmp/bacula-restores
}

# List of files to be backed up
FileSet {
Name = "Full Set"
Include = { Options { signature=SHA1; compression=GZIP9 }
File = /
File = /usr
File = /home
File = /boot
File = /var
File = /opt
}

Exclude = {
File = /proc
File = /tmp
File = /.journal
File = /.fsck
...
}
}

Schedule {
Name = "WeeklyCycle"
Run = Level=Full 1st sun at 2:05
Run = Level=Differential 2nd-5th sun at 2:05
Run = Level=Incremental mon-sat at 2:05
}

# This schedule does the catalog. It starts after the WeeklyCycle
Schedule {
Name = "WeeklyCycleAfterBackup"
Run = Level=Full sun-sat at 2:10
}

# This is the backup of the catalog
FileSet {
Name = "Catalog"
Include { Options { signature=MD5 }
File = /home/bacula/working/bacula.sql
}
}

Client {
Name = client-fd
Address = client
FDPort = 9102
Catalog = MyCatalog
Password = " *** CHANGE ME ***"
AutoPrune = yes # Prune expired Jobs/Files
Job Retention = 6 months
File Retention = 60 days
}

Storage {
Name = File
Address = localhost
SDPort = 9103
Bacula Version 7.4.4 265
Password = " *** CHANGE ME ***"
```

```
Device = FileStorage
Media Type = File
}

Catalog {
Name = MyCatalog
dbname = bacula; user = bacula; password = ""
}

Pool {
Name = Full-Pool
Pool Type = Backup
Recycle = yes # automatically recycle Volumes
AutoPrune = yes # Prune expired volumes
Volume Retention = 6 months
Maximum Volume Jobs = 1
Label Format = FullMaximum
Volumes = 9
}

Pool {
Name = Inc-Pool
Pool Type = Backup
Recycle = yes # automatically recycle Volumes
AutoPrune = yes # Prune expired volumes
Volume Retention = 20 days
Maximum Volume Jobs = 6
Label Format = IncMaximum
Volumes = 7
}

Pool {
Name = Diff-Pool
Pool Type = Backup
Recycle = yes
AutoPrune = yes
Volume Retention = 40 days
Maximum Volume Jobs = 1
Label Format = DiffMaximum
Volumes = 10
}

Messages {
Name = Standard
mailcommand = "bsmtp -h mail.domain.com -f \"\(Bacula\) %r\"
-s \"Bacula: %t %e of %c %l\" %r"
operatorcommand = "bsmtp -h mail.domain.com -f \"\(Bacula\) %r\"
-s \"Bacula: Intervention needed for %j\" %r"
mail = root@domain.com = all, !skippedoperator = root@domain.com =
mount
console = all, !skipped, !saved
append = "/home/bacula/bin/log" = all, !skipped
}
```

Appendix 4 Bacula Storage Daemon Configuration

```
Storage { # definition of myself
Name = bacula-sd
SDPort = 9103 # Director's port
WorkingDirectory = "/home/bacula/working"
Pid Directory = "/home/bacula/working"
}

Director {
Name = bacula-dir
Password = " *** CHANGE ME ***"
}

Device {
Name = FileStorage
Media Type = File
Archive Device = /files/bacula
LabelMedia = yes; # lets Bacula label unlabeled media
Random Access = Yes;
AutomaticMount = yes; # when device opened, read it
RemovableMedia = no;
AlwaysOpen = no;
266 Bacula Version 7.4.4 }
Messages {
Name = Standard
director = bacula-dir = all
}
```