Slovak University of Technology

Faculty of Informatics and Information Technologies

Team Project

NetCell – Management Document

Bc. Andrej Mlynčár

Bc. Martin Janočko

Bc. Tomáš Hermánek

Bc. Tomáš Mikuška

Bc. Lukáš Kleščinec

Bc. Ľubomír Kaplán

Academic Year: 2015/2016

Degree Course: Software Engineering

Table of Contents

List of Tables

List of Figures

1 Project Introduction

1.1 Project Goal

Goal of this software project is creation of network topology management web-based interface for OpenStack cloud computing system. Graphical interface will contain feature, which can help user to design and draw custom network topologies. This topologies can be deployed to production environment just few in clicks. Network topologies will contain virtual network functions (VNFs) such as firewalls, load balancers, proxies and services (httpd, db, …). Deployed VNFs will be configurable trough web interface of this system.

Project will offer also other functionalities as template management and group configuration management which is described below in more details. All these functionalities offered via graphical web interface can significantly contribute to better network management. Because of this fact, network maintenance and design done by system and network administrators and designers, can be much more efficient and easier. Especially in huge networks with many (>100) components, this system can significantly help to understand how network is designed, and how various network components can be configured.

1.1.1 Proposed Functionality

- **Topology Template Management** - the user needs to be capable of creating a network topology template. The template contains network devices and their interconnections.
- **Topology Deployment** - as of user's decision to launch an instance of topology template, the template including all its VNFs and network connections is built on top of OpenStack and put in functional state. This topology can be later modified and reconfigured and also saved as a template.
- **Group Configuration Management** - user can manage configuration of multiple devices as a single group – e.g. multiple web servers provisioned in load-balanced environment will be maintained with the same configuration.

1.2 Members

Following table contains basic information about our team members with email address:

Table 1 Team members

Last name	First name	E-mail	Title
Halagan	Tomáš	tomas.halagan@gmail.com	Ing.
Hermánek	Tomáš	tomhermanek@gmail.com	Bc.
Janočko	Martin	martin.janocko.gih@gmail.com	Bc.
Kaplán	Ľubomír	castor@castor.sk	Bc.
Kleščinec	Lukáš	lukas.klescinec@gmail.com	Bc.
Mikuška	Tomáš	mikuska.tom@gmail.com	Bc.
Mlynčár	Andrej	a.mlyncar@gmail.com	Bc.

1.3 Contact

Following table contains basic information about team and about contact information:

Table 2 Team contact

Team Number	2
Team Name	VirtNET
Product Name	NetCell
TP Official Website	http://team02-15.studenti.fiit.stuba.sk/
E-mail	virtnet@castor.sk

2 Project Roles

There are two types of roles used on our project: **Management roles** and **Short Term roles**. Each type is described below in more details.

2.1 Management Roles

This type of role is selected by skills of each team member. This role cab be switched between team members, but is necessary to inform other team members and add new roles distribution to this document. Roles in our team are:

Table 3 Management roles

Team Member	**Role**
Tomáš Mikuška	Test Manager
Tomáš Hermánek	Risk Manager
Martin Janočko	Communication Manager
Lukáš Kleščinec	Documentation manager
Ľubomír Kaplán	Head architect, environment manager
Andrej Mlynčár	Quality and process manager

2.2 Short Term Roles

Short term tasks were performed in whole course duration not just in sprint. Following table is containing list of team members, which were temporary assigned to mentioned short term roles.

Table 4 Short Term roles

Web page manager	Andrej Mlynčár, Lukáš Kleščinec, Martin Janočko
JIRA manager	Ľubomír Kaplán, Tomáš Mikuška
Team meeting manager	Everyone
Confluence manager	Ľubomír Kaplán, Andrej Mlynčár
Scrum master	Lukáš Kleščinec, Martin Janočko, Ľubomír Kaplán

3 Application of Management Processes

3.1 Communication Management

Communication methodic which were used during project development are in detail described in chapter 4.4 Communication Standards**.** According to these methodic, team members were able to communicate via following methods and systems:

- **Email communication.** This communication was mostly used when we communicated with our team leader, Ing. Tomáš Halagan. Email form of communication is only used for sending secure information, such as passwords, private keys etc.. Email communication is never used for formal conversations and group communication between regular team members.
- **Social media communication.** This kind of communication was used at the beginning of the winter semester, when another communication systems and methods were not available. After the first two weeks of winter semester, communication through social media is used only for informal communication. In the concrete, we used facebook group chat to discuss some informal topics, such as minor task details, interesting events during meetings, task reminders etc.
- **JIRA communication.** JIRA is one of most important communication method used in our team. At the beginning of winter semester, project was created with name VIRTNET on faculty JIRA instance https://jira.fiit.stuba.sk. Detailed information about our JIRA methodic are available in chapter 4.7 Issue Tracker Standards. In addition to issue and task tracking, JIRA was used to communicate about tasks, bugs and problems. This kind of communication was realized by discussion in comment sections of various JIRA tasks.
- **Confluence communication.** Confluence is our custom wiki system. Every important information about our project and team management is stored in this system. Confluence was used to organize meetings, store information about our meetings and discussion about huge tasks. No team members encountered any problems during using this system. More information about confluence methodic can be found in our 4.5 Documentation and Manual Standards and it is also mentioned in our 184.4 Communication Standards**.**

- **Team meetings.** This type of communication was our most efficient type of communication. Meetings were organized regularly couple of times every week. We can split our team meetings into two categories:
    - *Regular team meetings* – Attendance was desired from all team members. In this kind of meetings, we discussed major team project tasks, sprints, risks, important decisions etc. This type of meeting was organized approximately every week.
    - *Minor team meetings* – In this kind of meeting, one or more minor tasks were discussed and solved. Attendance was required from members who were responsible for these tasks. Meeting was organized more often, approximately one or two times a week.

3.2 Planning Management

Chapter contains basic information, how our team managed to use planning management to define plans for winter and summer semester.

3.2.1 Winter Semester

Main project objectives and goals are in detail described in our **engineering document – global goals for winter semester.** To achieve these objectives, following plan for winter semester was created at the beginning of the academic year. Plan was created on team meeting 29.09.2015.

Table 5 Project plan for winter semester 2015/2016

Date	Description
1.10.2015 – 12.10.2015	Creation high level architecture design. Creation of team standards. Communication and development systems setup (confluence, JIRA, gitlab).
1.10.2015 – 1.12.2015	Development environment (Openstack) for our software product setup.
November 2015	TP Cup – decision if we attend this competition.

12.10.2015 – 19.10.2015	Execution of first sprint: • Webpage creation. • Creation of first version of nodeconfig core. • Creation of first version of nodeclient core.
20.10.2015 – 2.11.2015	Execution of second sprint: • Tuning nodeconfig and nodeclient core systems, if changes are requested or missing functionality is found. • Creation of first configuration definitions. • Creation of definition testing tools.
3. 11. 2015 – 9.11.2015	Execution of third sprint: • Creation of client nodes scripts. • Tuning nodeconfig and nodeclient core systems, if changes are requested or missing functionality is found. • Setup of development environment (Openstack). • Documentation and webpage changes based on review from team leader.
10.11.2015 – 18.11.2015	Execution of fourth sprint: • Finalization of documentation for the 1st checkpoint. • Management core detailed architecture design. • Management core development – basic project setup. • Setup of development environment (Openstack)
18.11.2015	Submit first document to AIS.
20.11.2015 – 4.12.2015	Execution of fifth sprint: • Management core development – database, configuration, logging setup, web interface. • Node client development – launching of VNF service and basic configuration implementation.
5.12.2015 – 11.12.2015	Execution of sixth sprint:

	• Creation of final documentation for WS. • Node clients development – addition another basic configurations. • Management core development – web interface and login service implementation.
11.12.2015	Submit documentation and project prototype.

3.2.2 Summer Semester

Based on successful and unsuccessful goals defined in winter semester, following plan was created to achieve completion of desired product:

Date	Description
29.2.2016 – 16.3.2016	Execution of first sprint in this semester: Creation of basic configuration for node types. Development finish of components which were not finished in previous semester – Code Foundation components.
16.3.2016 – 10.4.2016	Execution of second sprint in this semester: Development of Node Type Clients with basic configuration. OpenStack integration with Management Core platform.
10.4.2016 – 1.5.2016	Execution of third sprint in this semester: Topology Management development. Complex Testing of developed VNF clients. Deployment of first production into UAT environment. Development finalization.
1.5.2016 – 20.5.2016	Finalization of team project. Presentation to product owner. Review of test result.

	Bug fixing and adding missing functionality. Creation of product documentation.

3.3 Documentation Management

Documentation methodic standards are described in chapter 4.5 Documentation and Manual Standards. At the beginning of the winter semester, confluence wiki was created, where every documentation part, external documents and attachment are stored. Confluence also provides document versioning, so we can keep tracking what changes were made to every document stored in our wiki.

Significant part of our fourth sprint was focused on creation of first version of management and engineering document. These two documents were successfully created and submitted to academic information system.

Creation of final version of documentation in winter semester was created separately, out of any sprints at end of the semester, after prototype was created. Some of documents, like team meeting reports, were created regularly. All of these documents were merged into one document and submitted in academic information system.

3.4 Risk Management

Risk management is one of the most important processes, which has to be handled to be efficient and successful in team project. At the beginning of the winter semester, risk management methodic were created. These methodic are documented in 4.9 Risk Management Standards. In the first 9 weeks of semester, we discussed and planned managing following risks:

Table 6 Risk - Absence of a team member on a meeting

Risk name	**Absence of a team member on a meeting**
Description	Team member is not able to attend team meeting.
Impact	Low
Probability	Medium

Trigger	Unexpected events can happen and team member is not able to attend team meeting.
Prevention	Team meeting date and time needs to be established with agreement of all team members.
Consequences	Team member thoughts, knowledge and commends are not available at a meeting.
Solution	Member who missed meeting has to review all tasks and notes which were created during meeting. He needs to approve these tasks and notes. In case he disagrees, disagreements will be discussed on next meeting.

Table 7 Risk - Team member is not able to finish a task

Risk name	**Team member is not able to finish a task.**
Description	Team member is not able to finish assigned task in time because of the incorrectly estimated task, unexpected problems or missing knowledge.
Impact	Medium
Probability	Medium
Trigger	Task estimation was set incorrectly. Bad time management from team member. Unexpected events can happen (sickness, school/work deadlines).
Prevention	Regular team reports - every team member needs to inform other members about progress on every assigned task
Consequences	Sprint is not completely finished. Task is moved to next sprint. Other team members have to finish uncompleted task.
Solution	Task will be distributed to other team members. In case task was not completed because of the bad time management, team member who was not able to complete task, will have more tasks assigned in next sprints.

Table 8 Risk - Unavailability of a team member for a long period of time

Risk name	**Unavailability of a team member for long period of time.**
Description	Team member is not able to work on team project due to illness, injuries etc.
Impact	Critical
Probability	Low
Trigger	Illness, injury, unexpected study ending.
Prevention	Avoid dangerous situations where team member can be injured or can get seriously ill.
Consequences	Plan created and project goal cannot be fully completed.
Solution	Risk cannot be fully handled. In case this issue will be occurred, work will be distributed between team members, project objectives will be reduced and project plan will be modified.

Table 9 Risk - Not possible to setup development environment

Risk name	**Not possible to setup development environment.**
Description	Openstack environment setup will not be possible.
Impact	Critical
Probability	Medium
Trigger	Missing hardware, unexpected technical problems.
Prevention	Make big effort to obtain requested hardware.
Consequences	Product development would be much more difficult, because of the missing environment, where product can be tested.
Solution	Openstack environment will be setup on virtual environment.

Table 10 Risk **-** Long term outage of a management system

Risk name	**Long term outage of a management system.**
Description	JIRA, Confluence or gitlab systems are not available for long period of time.
Impact	Critical
Probability	Low
Trigger	Server crash, database corruption, networking problems.
Prevention	Establish backup server which will be started in case of unexpected problems on main server.
Consequences	Team communication and scrum development will be significantly restricted. All tasks and team communication will be handled by other, nonconventional ways – email communication, facebook chat, other not well known system etc.
Solution	-

Table 11 Risk - Impossible to finish working prototype or documentation

Risk name	**Impossible to finish working prototype or documentation.**
Description	It will be impossible to submit working prototype to a date, when product should be submitted to academic information system.
Impact	Critical
Probability	Low
Trigger	Bad time management, low effort.
Prevention	Regularly work on product and documentation.

Consequences	Team project courses will not be successfully finished with good grades.
Solution	-

3.5 Development Management

During development of our products, we primarily focused on management processes described in this chapter.

3.5.1 Source Control

As it is described in 4.13 Version Control Standards, source control of programming projects was realized via cloud git repository management system gitlab. Access gitlab was granted at the beginning of academic year. Every team member except Ľubomír Kaplán have developer access rights, Ľubomír Kaplán is administrator of gitlab system, so he has unlimited access to every repository. Every team member have experience with operating git system, so only minor technical difficulties with initialization of repositories and accounts settings occurred, but they were quickly resolved.

Following repositories were created where source control of java, bash and python projects are managed:

- FIIT TP 2015 - VirtNET / netcell-nodeclient-core
- FIIT TP 2015 - VirtNET / netcell-management-portal
- FIIT TP 2015 - VirtNET / netcell-management-server
- FIIT TP 2015 - VirtNET / netcell-management-server-testbed
- FIIT TP 2015 - VirtNET / netcell-design
- FIIT TP 2015 - VirtNET / netcell-nodeconfig-core
- FIIT TP 2015 - VirtNET / netcell-nodeclient-webserver
- FIIT TP 2015 - VirtNET / netcell-nodeclient-openvpnserver
- FIIT TP 2015 - VirtNET / netcell-nodeclient-databaseserver
- FIIT TP 2015 - VirtNET / netcell-nodeclient-applicationserver
- FIIT TP 2015 - VirtNET / netcell-nodeclient-nameserver
- FIIT TP 2015 - VirtNET / netcell-nodeconfig-tools
- FIIT TP 2015 – VirtNET / sndb

3.5.2 Code Review

Code review standards are documented in 4.4 Communication Standards**.** To realize successful code review, status *"In Review"* was created in our custom JIRA scrum board. Every task which was transitioned to this status, had to be reviewed. Statistically, every task was reviewed in less than 24 hours after task was moved to status *In Review*. No problems were occurred with communication and distribution of tasks that needed to be reviewed. If some code corrections were requested from person who reviewed a task, corrections were made successfully before end of a sprint.

3.5.3 Issue Tracker

For issue tracking and task managing, JIRA system is used. Most of the team members already had basic experience with this system, so no problems were encountered with using JIRA. Every team member was able to follow issue tracking methodic described in chapter 4.7 Issue Tracker Standards. There were only some technical difficulties when we were not able to access JIRA, because of the issue with LDAP (Lightweight Directory Access Protocol) service. For minor tasks and issues Confluence is used. Typically, they are assigned during team meetings and subsequently documented into Confluence. After being resolved or fulfilled they are also marked so. List of all tasks from Confluence is available in chapter Task Export.

3.5.4 Backlog

Issues and tasks were created into backlog in our JIRA VNET scrum board. Most of the tasks were created to backlog during team meetings. Every team member has permission to create tasks to backlog. Issues which were not created during meeting were discussed and then approved or rejected during next meeting. Before first documentation submission there were 106 tasks and subtasks created in backlog, most of them were resolved in first 4 sprints. Before final winter semester document submission there were 166 tasks and subtasks created in backlog of which 150 were resolved.

4 Used Standards

List of used standards:

- 4.1 Backlog Standards,
- 4.2 Code Review Standards,
- 4.3 Coding Standards,
- 4.4 Communication Standards,
- 4.5 Documentation and Manual Standards,
- 4.6 External Document Storage Standards,
- 4.7 Issue Tracker Standards,
- 4.8 Linking Task and Issues Standards ,
- 4.9 Risk Management Standards,
- 4.10 Story Point Standards,
- 4.11 Team Meeting Standards,
- 4.12 Testing Standards,
- 4.13 Version Control Standards.

4.1 Backlog Standards

Backlog is one the most important things in entire development process. Backlog contains all tasks, which have to be done. From these tasks each sprint is defined.

4.1.1 Creating Backlog

Each team member has access to create issue in JIRA. This issue is added to backlog automatically. Our team uses four types of issues, from which everyone has specific characteristics:

- **story** – basic issue type for creating new part of system or new features etc. This issue type is mostly discussed by whole team on team meeting and then is created in backlog.
- **epic** – epic is used as group of more stories, which have common topic. Epic is used when story is too complex and is necessary to split him. This issue type is created same way as story.

- **bug** – bug is created when some team member (or user) finds functional defect in system behavior of created system. Each team member must create bug when he finds some misbehavior in application functionality.
- **task** – task is created when there is request to do something which is not included in development of software project (change in environment, change or setup of web page, etc.).

Each issue can be created on team meeting, where each team member can present his idea for work.

4.1.2 Backlog Issues Approving

As described in previous chapter, most backlog issues are created on team meetings, where we discuss problems of our product. Each backlog issue is then approved or rejected by team members. Whole team has to decide which issue is placed to the next sprint. After this whole sprint backlog is created, story point evaluation follows - this is described in chapter 4.10 Story Point Standards in more details.

4.2 Code Review Standards

Every submitted task needs to be reviewed. Code is always tied to a specific task. First reviewer should be the author of the code himself. After this, when the code is submitted, related task is moved to “In Review” status. Process and quality manager is responsible for informing another team member, that submitted task needs to be reviewed.

Team member is then responsible to provide a code review. In case he finds an issue in the submitted code, he needs to inform the original author of the code. It is necessary to mark the specific occurrence of the bug by line number or numbers and to write a short description of the error. Given team member should also provide a steps on how to reproduce the discovered error. After that the task is moved back to “In Progress” status and to the original author.

This process repeats until the task passes the review. One task is not limited for only one reviewer and it is advised to use more people for reviewing, especially if the submitted code is large.

4.3 Coding Standards

These standards are set of guidelines for programming languages that recommend programming style, methods and practices. Compliance with them allows us to create scalable and easy understandable code.

In our project, we use programming languages such as Java and Python. In both languages, different standards are used. Python uses PEP 8 convention.

4.3.1 Class and Interface Names

In Java – class and interface names begin with an uppercase letter and should not be pluralized unless it is a collection class. If it is multi-word name, *camel-case* style must be used.

```
class NetworkNode { }
```

In Python – class names should use the *CapWords* (similar to *camel-case*) conventions. Python do not use interfaces.

```
class NetworkNode():
```

4.3.2 Variable and Function Names

In Java – variable and function names starts with lower-case letter. Again, if it is multi-word name, *camel-case* style is used.

```
int currentNodeCount = 10;
```

In Python – lowercase with words separated by underscores are used (Snake-case).

```
current_node_count = 10
```

4.3.3 File Names

In Java - class and interface files use same name as class or interface inside them. Other files uses words separated by dashes.

```
nb-configuration.xml
```

In Python – file names consists from words separated by underscores.

```
netcell_nodeclient_core.py
```

4.3.4 Source Code Formatting

In our project, we are using four spaces as tabulator size. This standard is same for Java and Python.

In Java – opening curl braces starts at end of last command. End curl braces indented as well as block indentation. For further information about formatting, please read Oracle documentation.

In Python – curl braces is not used. Code block is determined by indentation. For further information about formatting, please read Oracle documentation.

4.3.5 Source Code Commenting

Java uses 4 styles of comments.

- Block Comments – they are used to provide descriptions of files, methods, algorithms and data structures.

```
/*
* Block comment example
*/
```

- Single-Line Comments – it appears on single line intented to the level of code that follows.

```
/* Single-Line comment example */
```

- Trailing Comments – short comment at same line as code.

```
return true;              /* description of this case */
```

- End-Of-Line Comments – comment begins a comment that continues to the newline.

```
return true;              // Explanation begins here
// something
// interesting
// here
```

Python uses 3 styles of comments:

- Block Comments - block comments apply to some code that follows them, and are indented to the same level as that code.

```
'''
Block comment example
'''
```

- Inline Comments – it is on the same line as a statement.

```
return True               # Explanation
```

- Documentation Strings – they are used in docstring.

```
"""
```

```
Doc string example
“””
```

4.4 Communication Standards

Multiple systems are used for communication within the team. As a primary language was chosen English language. This language is used in every official communication channel. In the informal channels the language is not important, but Slovak language is preferred. Based on the importance of the topic and formality you should use one of the following systems for communication.

4.4.1 Social Media Communication

Social media communication represents the informal part of communication between the team members. If there is a topic not tied to any specific task or the topic needs to be communicated quickly than this is the preferred communication channel. This channel should be also used in case email or personal communication is not possible.

4.4.2 Email Communication

Any topic that requires attention of the team leader should be communicated through emails. Besides team communication, emails communication is used for distribution of login credentials to various systems that the team members will have access to. This communication channel is also preferred in case the communication needs to be stored for longer period of time. If writing an email, it is necessary to include all the team members and also team leader.

4.4.3 JIRA Communication

JIRA is used to track all the issues and tasks regarding the project. Because of this, communication regarding specific tasks should be discussed right below the issues in the comments. This includes all the problems like mistakes in the attached documents and bugs in related code sections. Comments need to be short and to the point. Comments also enable a short discussion for the related task. If more information is needed to be posted, then it is preferred to store it in Confluence and link it with the specific issue or task.

4.4.4 Confluence Communication

Confluence is the team wiki system. All the information regarding our project should be stored here. This information include all the documents related to project like specifications, documentation and meeting write ups.

4.4.5 Personal Communication

Personal communication is used during team meetings. These need to be organized regularly. Team meetings, as a part of formal communication, do not need to be held in English language. Every meeting needs to be documented and this document needs to be stored in the Confluence. This document has to be written in English. Personal communication also includes informal communication outside of the team meetings.

4.5 Documentation and Manual Standards

4.5.1 Documentation Building Process

Every part of documentation will be created on local computer of team member. Each document has to be created according to documentation template which is included in confluence “Documentation for Team Project course” page.

Parts of documentation are defined by following source:

http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/materialy/dokumentacia2015-16.pdf

Each part has its own JIRA ticket. This ticket is used for versioning and correction in "review process". Each created document is attached to his issue and sent to review by other team member. Versions of documents in review will be attached to JIRA issue in each review iteration. Final version of document will be uploaded into confluence page of checkpoint documentation (child page od “Documentation for Team Project course”).

4.5.2 Documentation Styling

Don’t use "Heading 1" when writing part of documentation, this will be used only in final version of checkpoint document.

Heading 1:

- font: Arial,

- size: 20,
- alignment: justify,
- line spacing: 1,5 line,
- used heading numbering.

Example: 1 Heading 1

Heading 2:

- font: Arial,
- size: 18,
- alignment: justify,
- line spacing: 1,5 line,
- used heading numbering.

Example: 1.1 Heading 2

Heading 3:

- font: Arial,
- size: 16,
- alignment: justify,
- line spacing: 1,5 line,
- used heading numbering.

Example: 1.1.1 Heading 3

Heading 4:

- font: Arial,
- size: 14,
- alignment: justify,
- line spacing: 1,5 line,
- used heading numbering.

Example: 1.1.1.1 Heading 4

Heading 5:

- font: Arial,
- size: 12,
- alignment: justify,
- line spacing: 1,5 line,
- used heading numbering,
- used bold highlight.

Example: **1.1.1.1.1 Heading 5**

Paragraph:

- font: arial,
- size: 12,
- alignment: justify,
- line spacing: 1,5 line,

Example:

This is example text: Lorem ipsum dolor sit amet., consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Pictures and tables:

- alignment: center,
- title:
  - position: under picture,
  - alignment: center,
  - font: Arial,
  - size: 9,
  - picture number: bold,
  - picture description: without highlighting.

Example:

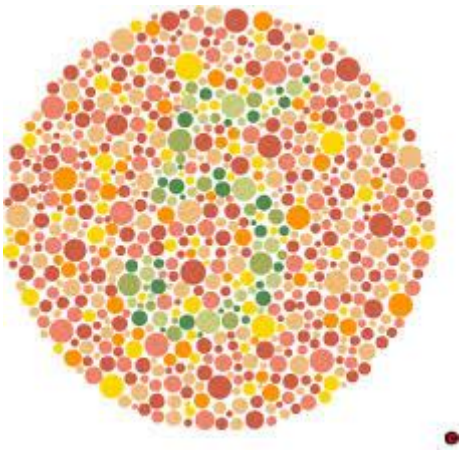

Figure 1 Test Picture

Indentation:

- first level: solid dot,
- second level: indent,
- third level: blank dot.

Example:

- first example,
- second example,
- second sub-example,
- second sub-example,
- very deep example.

4.5.3 User Manual Building Process

User manuals are divided into two different types of documents, which each have some specific rules and that is:

- user manuals,
- development and testing tools manuals.

Each type of manuals is described below in more details. Styling of each manual is the as styling of documentation and is described in chapter 4.5.2 Documentation Styling.

4.5.3.1 User Manuals

User manuals need to be included in final version of documentation, because these manuals are very important for users of our system. They cover whole system functionality with user guidelines. Before user manual is created, there have to be

raised JIRA ticket, which contain request for creating of manual (for more information see chapter 4.1 Backlog Standards). Manuals will be created in confluence in “NetCell Documentation” space. User manual must contain following information:

- prerequisites,
- step-by-step instruction for desired result,
- screenshots with description (when about GUI is writing).

4.5.3.2 Development and Testing Tools Manuals

This type of manuals doesn’t needs to be included in final version of documentation, or presented on web page. These manuals are created for development and test purposes, so they are addressed for team members only. Every development manual has to be present in confluence page of concrete tool or specific software for developers. These manuals writing when new tool or methodology is created or changed.

Development and testing manual must contain information about tool purpose and all information which are necessary to correct using of tool:

- prerequisites – OS, platform, compiler, etc.,
- instruction to run tool,
- tool options,
- tool results.

4.6 External Document Storage Standards

As external documentation storage we use team collaboration system Confluence. Confluence consists of two main document spaces.

Main space is called *NetCell Development.* This space has to be visible only for team members.

NetCell Development. Internal document space. Every important information about development needs to be stored into this document space. Every team member has permission to create additional web page into this space. Content needs to be categorized based on system module hierarchy, components and development branches.

NetCell Documentation. Public document space. Space needs to contain product public documentation, which can significantly help NetCell product users understand basic concepts of our system.

Before storing new document to external storage, proper place must be chosen. If document topic is not qualify for any part of the main page, new one can be created. Creation of new space needs to be approved by majority of team members.

4.7 Issue Tracker Standards

Our team use JIRA system software as our primary issue tracker system. Our team JIRA is available on web page www.jira.fiit.stuba.sk. Each member has access rights to JIRA which permits to create an issue. Lukáš Kleščinec, Andrej Mlynčár and Ľubomír Kaplán has also administrators rights. Our team use own JIRA project named VirtNet (VNET), which have custom workflow with following states:

- **TODO** – first state of created issue. Issue in this state is without assignment or the work has not started yet.
- **IN PROGRESS** – some team member is working on issue, when issue is in this state.
- **IN REVIEW** – issue in this state is solved, but another team member must check work progress and he can return issue to IN PROGRESS state.
- **DONE** – all work on issue was done correctly.

This workflow is used in combination with Agile Scrum Board which is plugin for agile software development method. Agile Scrum Board allows creation of sprints from created issues. Each issue in sprint is estimated by story point which is described in chapter 4.10 Story Point Standards in more details. Agile Scrum board also creates burndown charts of each sprint (see more in chapter 5.2 Sprint summary).

Each Jira issue is created with specific priority. This priority can be changed and it helps to effectively work on issues. We used following priorities:

- Trivial – these kinds of issues are usually added to sprint at last,
- Minor – this problem does not need a quick resolution,
- Major – major level of importance,
- Critical – very high level of importance,

- Blocker – blocks further development of project, should be resolved as quickly as possible.

Each issue can be assigned to specific component of our software product. We used our custom components described in following table:

Table 12 JIRA Components

Component	Component Lead
Management Server	Ľubomír Kaplán
NT Application Server	Tomáš Mikuška
NT Database Server	Tomáš Mikuška
NT Nameserver	Lukáš Kleščinec
NT Node Client Core	Lukáš Kleščinec
NT OpenVPN Server	Tomáš Hermánek
NT Web Server	Martin Janočko
Others	Andrej Mlynčár

In case system development will contain new area of interest, new component can be created. Every component needs to have component leader.

Issue created with specific component will be assigned to component leader. Based on team meeting discussion and sprint specification this issue can be later assigned to another team member. Issues which do not have component are assigned on team meeting after proper discussion.

Work on issue must be logged into JIRA. Each team member must log his work with date when he starts work on issue.
Sometimes, we create some minor, unimportant tasks in team meeting report. These tasks cannot be reviewed because output is not documented. For instance, study of documents can be topic of this kind of task.

4.8 Linking Task and Issues Standards

Every sprint contains issues, which are picked up from backlog. Process of creating tasks is very important for successful project completion. Each task has assigned story

points, which are voted by team members. As scale, we are using Fibonacci sequence. Manager of quality is responsible for linking tasks.

4.8.1 Issue Types

JIRA allows us to create some basic issue types.

- Bug - problem which impairs or prevents the functions of the product,
- Task – task of unspecified type, which must be done,
- Story – issue type for user story,
- Epic – issue type that needs to be broken down to multiple stories and usually takes longer than one sprint to get resolved.

4.8.2 Linking

Linking allows creation of associations between two existing issues. Also, it is possible to make many-to-many association between tasks, but between two specific issues only one link can be created. For example, basic link types are listed below:

- Relate – an issue may relate to another,
- Duplicate - an issue may duplicate another,
- Block – an issue may block another (is task is blocked by other issues, it cannot be resolved before issues which are blocking it!).

Also, it is possible to create association link between JIRA issue and Confluence page. We are using this option in our project.

Tasks, Bugs and Stories can be part of an Epic. This is called Epic link.

We are linking issues anytime, when it is appropriate. If task is closed, it should not be linked with another task, because if issues are linked, we expect that some work will be done on both of them.

4.8.3 Task Status

When sprint starts, each issue has a "*To Do"* status. After that, team member picks up the task and tries to solve it. Issue can be in one of the following states:

- "To Do" – not started task,
- "In Progress" – task, which is currently being resolved,
- "In Review" – resolved task, which waits for review by other team members,
- "Done" – resolved and closed task.

4.8.4 Task Priorities

Priority issue level indicates its importance. We are using these priorities in the project:

- Blocker – blocks further development of project, should be resolved as quickly as possible,
- Critical – very high level of importance,
- Major – major level of importance,
- Minor - this problem does not need a quick resolution,
- Trivial – these kinds of issues are usually added to sprint at last.

4.9 Risk Management Standards

The main role of risk management is to reveal potential risks, which can arise during creation of software product. After, that it is good practice to propose steps, which can reduce potential risks. These steps are essential to successful project completion.

4.9.1 Unrealistic Plans

Probability of this risk is high in a newly established team or after acceptance of a new team member. This risk is one of the most frequent risks. Our team members have different schedules. This causes that tasks are usually solved at the last moment.

To minimize risk, tasks are selected from backlog to sprint for participation by all team members. Issues are evaluated by story points. As scale we are using Fibonacci sequence. Once we have agreed on an assessment tasks, two-week sprint is created.

Risk is also minimized by fact, that the team has the same lineup.

4.9.2 Absence of a Team Member on a Meetings

Probability of this risk is medium but still possible. Our team members have different schedules. This causes that the team member can absence on team meeting.

To prevent this, team members private schedule is shared between other members and through communication canals (especially chat), we are able to plan meeting together.

Member, who missed meeting, must read team meeting reports and discuss with other members about it.

4.9.3 Team Member is not able to Finish Task

The team consists of people, which has different knowledge. Therefore some tasks are hard to evaluate correctly by all team members. Also, task can be assigned to member who doesn't have enough knowledge.

On team meetings is crucial that, the all members selects tasks according to their expertise.

In our team, members help each other with the definition of the problems. Task is also possible to be distributed between other members.

4.9.4 Unavailability of Team Member for Long Time Period

Team members works on different modules. This causes that some of us are insufficiently aware of all the components. If team member which has a unique understanding of the implementation of specific component becomes ill, there is possibility that the task can't be done.

To remove this risk, we must on each team meeting discuss in detail all the tasks. Then, team is able to distribute task between other members.

4.9.5 Conflicts in Resolving Tasks

This risk occurs when there are some tasks, which overlaps to each other. There may be various problems such as conflicts in linking code or unnecessary duplication work.

To minimize this risk, is important good communication between team members. We are using many communication channels, which helps us to solve conflicts correctly.

4.9.6 Not Possible to Setup Development Environment

Team product, will be installed on OpenStack environment. At this time, we have no access to this environment, but we are still developing software product.

We are not possible to minimize this kind of risk, but as backup plan, software product will be set up on virtual environment.

4.9.7 Long Term of Outage of a Management System

There is possibility that, Gitlab, Counfluence or Jira will be disabled for long time period. Working process will be significantly restricted, because we will be forced to use nonconventional ways – email, social networks, etc.

To prevent this, it is good practice to use many different systems, which can substitute others.

4.9.8 Impossible to Finish Working Prototype or Documentation

As students, we are mandatory to submit working prototype or create documentation up to date. After that, team project courses will not be successfully finished with good grades.

Solution is to set up realistic plans.

4.10 Story Point Standards

Agile development is based on a repeating events called sprints. Prior to each sprint it is necessary to create tasks that are going to be completed during the upcoming sprint. These tasks are also evaluated based on their difficulty. Unit of a difficulty is called Story Point.

4.10.1 Story Point Scale

There are multiple methods of evaluation. Our team has decided, that we will use an evaluation with slight modification of numbers from Fibonacci's sequence. Particularly 1, 2, 3, 5, 8, 13, 21, 40, 100. Number 1 represents easily completed task, 13 difficult task and number 100 represents currently impossible task. In case task Story point value is higher than 13, e.g. 21 and more, this task is split into multiple subtasks.

Before the evaluation starts, one of the tasks is chosen as a reference task. Team agrees on how much Story Points it is worth. Optimal value should be in the lower half of the Story Point scale. This task will then become a reference task based on which the others will be evaluated. Every other task should be compared against this one.

4.10.2 Task Evaluation

Tasks are evaluated one by one. Each member of the team can decide on his own how many Story Points worth of work belong to each task.

Story Points assignment goes as following:

1. Team decides on a reference task.
2. Reference task is assigned Story Points value.
3. Scrum Master chooses next task being evaluated from project backlog.
4. Team members show their evaluation of the task, all at the same time.
5. If everyone agrees on a common amount of Story Points, these are then assigned to the task.
6. If there are differences, team members explain their reasoning.
7. Voting repeats until there's an agreement.
8. Process from step 3 repeats until there are no tasks or the Story Point limit for sprint has been reached.

4.11 Team Meeting Standards

Team meetings are divided into two types of meetings (as is mentioned in document Application of Management Processes):

- regular team meetings,
- minor team meetings.

4.11.1 Regular Team Meetings

This type of meetings is for each team member main and attendance of each team member is mandatory. Time of this meeting is set on Tuesday at 11:00 AM and this meeting is organized every week. Each team member can request to change team meeting time via social media communication. If all other team members agree on changed team meeting time, following meeting is in this time.

This meeting is used to discuss most important things in and major tasks. During this meeting sprint backlog is created (for more details see chapter 4.1 Backlog Standards) and tasks in backlog are estimated (for more information see chapter 4.10 Story Point Standards).

This meeting must be documented (see chapter 4.11 Team Meeting Standards).

4.11.2 Minor Team Meetings

This type of team meetings is not mandatory for each team member. Attendance is required from members who are responsible for tasks discussed on meeting. Other

team members can be present on this meetings too. Minor meetings are organized for discussion of some tasks in backlog with members who are responsible for these tasks. After resolution, other team member will be informed about changes or updates in particular system part.

There is no obligation to create documentation from these meetings, but it is recommended in case some important issue is discussed or being resolved.

4.11.3 Team Meeting Documentation Standards

As shown in previous chapter mostly regular meetings are documented. On each meeting inception one team member is chosen, who is responsible for meeting reporting. Team meeting report is written into Confluence using our custom template for meeting reports. Each meeting report contains the following information:

- **Attendance review** – names of present team members,
- **Primary topic** – just short description of main meeting topic,
- **Goals** – goals are most important points of each team meeting. Whole team must discuss each goal (sprint summary, new sprint, important decisions and etc.) and agree on common solution.
- **Notes** – important notes. They are found when team discusses some goal.
- **Tasks** – created tasks are based on notes and goals finds on meeting. Each task must be assigned to one or more team members.

4.12 Testing Standards

The main role of test management is to reveal potential bugs, which can arise during creation of software product. After that, it is good practice to propose steps, which can cover the biggest amount of possible bugs. These steps are essential to successful project completion.

4.12.1 Test Creation

Every team member must analyze the desired functionality and design individual tests, which needs to cover the most important parts of system module.

Developer of system module or system part must implement the required tests based on the design. These steps can be executed before or after the tested functionality is implemented.

4.12.2 Unit and Integration Testing

Unit and integration tests are designed, created and executed by system module developer, but not necessarily after part of source code is written. Tests must be focused on possible error arising during runtime. Most often the white-box testing has to be used for unit test creation, except the test lead recommends other testing type.

4.12.3 System Testing

System tests must be designed to verify non-functional requirements of as functionality, reliability, security, robustness, maintainability and performance. This kind of tests are not necessary for our kind of development. Tests can be created after ending of system development.

4.12.4 Acceptance Testing

Testing should be executed after a total system tuning and correction of all errors from previous tests. All the required tests and all functional requirements must be included in acceptance tests.

4.13 Version Control Standards

This methodic document describes rules and procedures, how to work with version control system Git, repository management GitLab and collaboration software Confluence.

4.13.1 Keywords

Git – version control system.

Repository – storage of specific files.

Branch – developing line.

Commit – record of changes in repository.

API – application programming interface.

Pull – fetch from integration with another repository or a local branch.

GitLab – Git repository server.

Confluence – documents and sources organization system.

4.13.2 Repositories

Whole project is divided into separate repositories, which are under administration of different team members. They are stored on GitLab repository server. At the same time, there is only one member, who is working on selected repository. Therefore, it is not necessary to create branches and deal with merging. All of them can be also divided into two groups. These repositories are described below.

4.13.2.1 Core Group

This group is under administration of Ľubomír Kaplán and Andrej Mlynčár. It covers creation of network topology management web-based interface for OpenStack cloud computing system. In present, there are repositories listed below.

- netcell-nodeconfig-core – collaborates with netcel-nodeclient-core for purpose of configuration of virtual machines,
- netcell-nodeconfig-tools – tools for verification of configurations,
- netcell-management-server – system core application,
- netcell-management-server-testbed – testbed for management server,
- netcell-design – design files related to project and user interfaces.

4.13.2.2 Node Client Group

This group is under the administration of Martin Janočko, Tomáš Hermánek, Tomáš Mikuška and Lukáš Kleščinec.

- netcell-nodeclient-core – API server for processing HTTP requests, every server which is listed below, uses it as sub-module,
- netcell-nodeclient-webserver – implementation of configuration for webserver service,
- netcell-nodeclient-openvpnserver - implementation of configuration for VPN service,
- netcell-nodeclient-databaseserver - implementation of configuration for database service,
- netcell-nodeclient-applicationserver - implementation of configuration for application server service,
- netcell-nodeclient-nameserver - implementation of configuration for name server service.

4.13.3 Development of New Functions

This section describes how to work with repositories, make commits, pull changes and push changes. For further information, please check Git documentation.

4.13.3.1 Update of Current Working Repository

Developer, who resolves the tasks from Jira, must download the latest version from the repository server. To achieve this, use Git command pull.

Example:

```
git pull origin master
```

4.13.3.2 Making Changes

Each meaningful change should be committed. One JIRA task can contain multiple changes. Commit should contain message in specific format, which includes JIRA issue ID and some description text message. Providing JIRA issue ID is mandatory in case the commit is resolving a bug.

Example:

```
git commit <file> -m 'VNET-777: login bug fixed'
```

4.13.3.3 Storing Changes

Developers must store changes to the GitLab server on regular basis, always after finishing work. To achieve this, use push command.

Example:

```
git push origin master
```

4.13.4 Documentation of New Functions

Each one of the new functionality must be documented and described in Confluence server. It allows the team to keep track of versions. For further information refer to Confluence website.

4.13.4.1 Document Organization

Document is organized into spaces. Currently using three spaces:

- NetCell Documentation – technical documentation of project,
- NetCell Development – technical information for developers.

Content of spaces is hierarchically organized into pages. Each repository has its own page, which contains:

- Overview – basic functionality overview,
- Downloads – some downloads links,
- Change Log – documents changes,
- Usage – information how to use sources,
- Examples – examples of usage.

4.13.4.2 Change Log

After resolution of an issue from JIRA, new version is created. If more than one task is part of scrum sprint, version can resolve more than one issue. Change log table contains four columns:

- Date – date of version creation,
- Version – version number,
- Build Number – implementation build number,
- Changes – description of changes in specific version, also contains Jira issue ID.
- Version, also contains Jira issue ID.

5 Task Export

Below is a task export from team meetings in a chronological order from the beginning of winter semester. JIRA tasks are located in chapter 5.2 Sprint summary.

5.1 Project Team Meetings

Tasks assigned during team meetings are supposed to supplement tasks assigned to sprints. Usually, these tasks and issues are not severe enough to be included into active sprint tasks or general tasks to be done.

5.1.1 Project Team Meeting on 22.9.2015

- Andrej Mlynčár, Martin Janočko - Study of bachelor project created by Dušan Matejka, which is focused on Virtualization and development of API for software Load Balancer - HAProxy.
- Lukáš Kleščinec, Tomáš Hermánek, Tomáš Mikuška - Review of their bachelor thesis which were also focused on Virtualization and development of API for software - Squid, IpFilter, OpenVPN.

5.1.2 Project Team Meeting on 29.9.2015

- Tomáš Halagan - Provide hardware for development.
- Andrej Mlynčár, Ľubomír Kaplán - Start work on concept of management server.
- Lukáš Kleščinec - Study for development of DNS server (bind9).
- Lukáš Kleščinec - Ensure access to JIRA (communication with Michal Barla).
- Tomáš Hermánek - Study for development of OpenVPN Server.
- Tomáš Mikuška - Study for development of database server (mysql).
- Tomáš Mikuška - Study for development of application server (glassfish).
- Martin Janočko - Study for development of web server (apache2 / nginx).

5.1.3 Project Team Meeting on 7.10.2015

- Create specification for node configuration and node configuration definition.
- Start development of node client core, which will load node client module (specific to node type).

5.1.4 Project Team Meeting on 8.10.2015

- Start first sprint named "Dub Phizix" on 12.10.2015 with duration of 1 week.

- Lukáš Kleščinec, Martin Janočko, Tomáš Mikuška, Tomáš Hermánek - Develop Node Client Core (API, API calls, module loading).
- Andrej Mlynčár, Ľubomír Kaplán - Develop Node Configuration Core library (entities, parser, validator, serializer, deserializer, navigator).
- Lukáš Kleščinec - Develop team website.

5.1.5 Project Team Meeting on 19.10.2015

- Tomáš Hermánek - Create basic configuration of OpenVPN Server
- Tomáš Mikuška - Create basic configuration of Database Server and Application Server
- Lukáš Kleščinec - Create basic configuration of Name Server
- Martin Janočko - Create basic configuration of Web Server
- Ľubomír Kaplán, Andrej Mlynčár - Specification and brainstorming of management server core

5.1.6 Project Team Meeting on 21.10.2015

- Tomáš Mikuška - Rework tests for netcell-nodeclient-core module.
- Andrej Mlynčár, Ľubomír Kaplán - OSGi study.
- Ľubomír Kaplán - Draw diagram for NetCell management server.

5.1.7 Project Team Meeting on 28.10.2015

- Andrej Mlynčár - Contact prof. Bieliková regarding the language of web page and documentation

5.1.8 Project Team Meeting on 3.11.2015

- Ľubomír Kaplán - Migrate ngnlab.eu machines and setup OpenStack.
- Martin Janočko - Translate webpage.
- Tomáš Hermánek - Create node client core automatic startup.
- Lukáš Kleščinec - Create Debian 8 Jessie image for OpenStack.
- Andrej Mlynčár - Modify node configuration core.
- Tomáš Mikuška - Complete node client API method.

5.1.9 Project Team Meeting on 9.11.2015

- Lukáš Kleščinec- Redistribute tasks for 1st checkpoint documentation.
- Andrej Mlynčár, Ľubomír Kaplán - Develop MS Core Foundation.

- Andrej Mlynčár, Ľubomír Kaplán, Lukáš Kleščinec, Tomáš Mikuška, Martin Janočko, Tomáš Hermánek - Discuss configuration bundle structure.

5.1.10 Project Team Meeting on 18.11.2015

- Andrej Mlynčár - Submit documentation into academic information system.

5.1.11 Project Team Meeting on 23.11.2015

No additional tasks were assigned during this meeting.

5.1.12 Project Team Meeting on 7.12.2015

- Tomáš Mikuška - Create Testing Standards.
- Tomáš Mikuška - Create presentation for Testing Management.
- Ľubomír Kaplan - Create presentation for Development Management.
- Lukáš Kleščinec - Create presentation for Documentation Management.
- Andrej Mlynčár - Create presentation for Quality Management.
- Martin Janočko - Create presentation for Communication Management.
- Tomáš Hermánek - Create presentation for Risk Management.

5.1.13 Project Team Meeting on 10.12.2015

No additional tasks were assigned during this team meeting.

5.1.14 Project Team Meeting on 22.2.2016

First team meeting in the summer semester. Team meeting was focused on creation of plan for the upcoming semester. Sprint duration was set for 2 weeks. There will be 3 sprints during summer semester.

5.1.15 Project Team Meeting on 29.2.2016

New sprint, named “Five Finger Death Punch”, started. Sprint will last from 1.3.2016 to 16.3.2016. Sprint contains 26 tasks evaluated at 95 story points. Main tasks assigned for this sprint:

- Tomáš Hermánek – VPN Basic Config,
- Martin Janočko – Webserver Basic Config,
- Ľubomír Kaplán – OpenStack Manager Service Implementation,
- Lukáš Kleščinec – NameServer Basic Config,

- Tomáš Mikuška – Application Server Basic Config,
- Tomáš Mikuška – Database Server Basic Config,
- Andrej Mlynčár – Identity Service Implementation.

5.1.16 Project Team Meeting on 7.3.2016

This team meeting was held in the middle of the sprint. Main reason for the meeting was, that there was too many tasks assigned. We went over unfinished tasks and prioritized those, who had to be finished.

5.1.17 Project Team Meeting on 16.3.2016

Sprint “Five Finger Death Punch” was finished. 17 tasks out of 26 were finished successfully. This means that 58 out of 96 projected story points were obtained. Tasks that were not completed during the last sprint will be transferred to the next sprint. Team managed to complete all important tasks.

5.1.18 Project Team Meeting on 21.3.2016

New sprint, named “DMS”, started. This sprint was running from 22.3.2016 to 11.4.2016. Unfinished tasks from previous sprint were also added. Sprint contains 20 issues evaluated at 53 story points. Following are the main tasks:

- Tomáš Hermánek – Create VPN Server post-install script and create basic functional tests for it,
- Martin Janočko – Create Webserver post-install script and create basic functional tests for it,
- Ľubomír Kaplán – OpenStack Manager Service Implementation, Netcell Task Manager implementation,
- Lukáš Kleščinec – Create Nameserver Server post-install script and create basic functional tests for it,
- Tomáš Mikuška – Create Application and Database Server post-install script and create basic functional tests for it,
- Andrej Mlynčár – Identity Service Implementation, Logging improvements.

5.1.19 Project Team Meeting on 4.4.2016

This team meeting was organized to review the ongoing sprint. All previously unfinished tasks were completed. The rest of the team meeting was dedicated to work on the tasks.

5.1.20 Project Team Meeting on 11.4.2016

Sprint “DMS” finished successfully. All of the tasks were completed. Team managed to get 53 out of 53 story points for 20 tasks assigned to this sprint.

New sprint, named “Arctic Monkeys” was started. There were 9 tasks assigned to this sprint. These tasks were evaluated at 49 story points. Following were the main tasks assigned during this sprint:

- Tomáš Hermánek – Create basic test cases for VPN Server,
- Martin Janočko – Create basic test cases for Webserver,
- Ľubomír Kaplán – Netcell nodetype manager development, OSGi refactoring,
- Lukáš Kleščinec – Create basic test cases for Nameserver,
- Tomáš Mikuška – Create basic test cases for Application and Database server,
- Andrej Mlynčár – Create testing environment.

5.1.21 Project Team Meeting on 18.4.2016

Evaluation of the ongoing sprint. There were 6 tasks out of 9 finished at this point. Development continued during the team meeting.

5.1.22 Project Team Meeting on 26.4.2016

Sprint “Arctic Monkeys” finished. All 9 tasks were finished. In total 49 story points were obtained.

5.1.23 Project Team Meeting on 5.5.2016

Team meeting dedicated to evaluate current state of the project. Final development continued during this meeting.

5.1.24 Project Team Meeting on 16.5.2016

Documentation of the team project was being created at this team meeting. Tasks were assigned as follow:

- Tomáš Hermánek – Create documentation for VNP server implementation,
- Martin Janočko – Create documentation for Webserver implementation,
- Ľubomír Kaplán – Create documentation for core services implementation,
- Lukáš Kleščinec – Create documentation for Nameserver implementation,
- Tomáš Mikuška – Create documentation for Application and Database server implementation,
- Andrej Mlynčár – Create static form of team webpage.

5.1.25 Project Team Meeting on 17.5.2016

Finalizing documentation.

5.1.26 Project Team Meeting on 18.5.2016

Finalizing documentation. Merging all documents.

5.2 Sprint summary

5.2.1 Dub Phizix

Number of sprint: 1
Name of sprint: Dub Phizix
Sprint date: 12/Oct/15 6:30 PM - 19/Oct/15 12:45 AM
Number of issues: 12
Number of story points: 30

The first sprint named Dub Phizix was primarily aimed at creation of node configuration core and node client core. Within the node configuration core module were implemented methods for creating, parsing, serialization and loading of configuration. Also were specified node configuration format and node configuration definition. Within the node client was created core module at which the server is run. Methods for getting node type information and providing node configuration definition were created.

During the first sprint was built webpage on which will be placed all the documents related to the work on team project. The webpage also contains basic information about project and team members.

Bad practices in issue solving were used during the first sprint as it is shown in Figure 2. Solved issue was moved to state “Done” even if not solved correctly. On the basis

of this problem new column “In review” which will be used for checking of solution correctness was added. Entire list of issues solved during the sprint is shown in Figure 3.

Figure 2 Dub Phizix burndown chart

Completed Issues

Key	Summary	Issue Type	Priority	Status	Story Points (30)
VNET-2	Node Client Core Foundation	Story	Major	DONE	3
VNET-4	Node Client API Server	Story	Major	DONE	2
VNET-5	Node Client Get Node Type Info API Method	Story	Major	DONE	2
VNET-6	Node Client Get Node Config Definition API Method	Story	Major	DONE	2
VNET-9	Node Config Definition Specification	Story	Major	DONE	1
VNET-11	Node Config Core - Parse Definition	Story	Major	DONE	5
VNET-12	Node Config Core - Create Default Config	Story	Major	DONE	3
VNET-13	Node Config Core - Serialize and Load Config	Story	Major	DONE	5
VNET-14	Node Config Core - Navigate Config	Story	Major	DONE	2
VNET-16	[WEB] Deploy Wordpress app on Virtual Server	Task	Minor	DONE	2
VNET-17	[WEB] Develop web v 0.1	Task	Minor	DONE	3
VNET-19 *	Setup Confluence	Task	Major	DONE	-

* Issue added to sprint after start time

Figure 3 Dub Phizix issue list

5.2.2 Pink Floyd

Number of sprint: 2

Name of sprint: Pink Floyd

Sprint date: 19/Oct/15 5:50 PM - 02/Nov/15 5:09 PM

Number of issues: 17

Number of story points: 46

The second sprint named Pink Floyd was primarily aimed at creation of the first client configuration definitions and their verification by definition testing tools. Also new configuration types such as group, list, password, options and byte size were defined.

By the adding of new "In Review" column to improve the issue solving, the red line of remaining values took excepted downward shape, as it is shown in Figure 4. Column is used for solution correctness checking, so it cannot happen that the issue is closed without its review. On the other side, significant part of issues were solved at the end of the second sprint. The main reasons of this problem were bad team communication and sprint length. During the sprint every issue was solved, as it is shown in

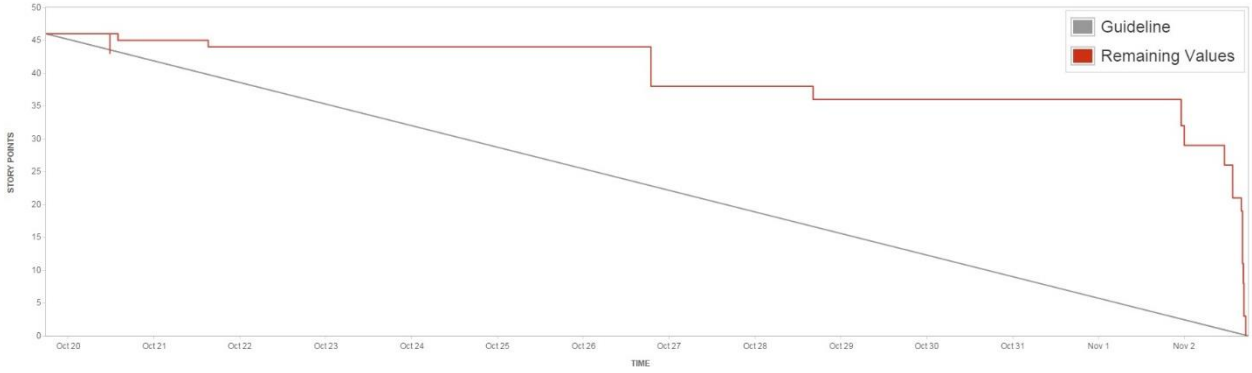

Figure 4

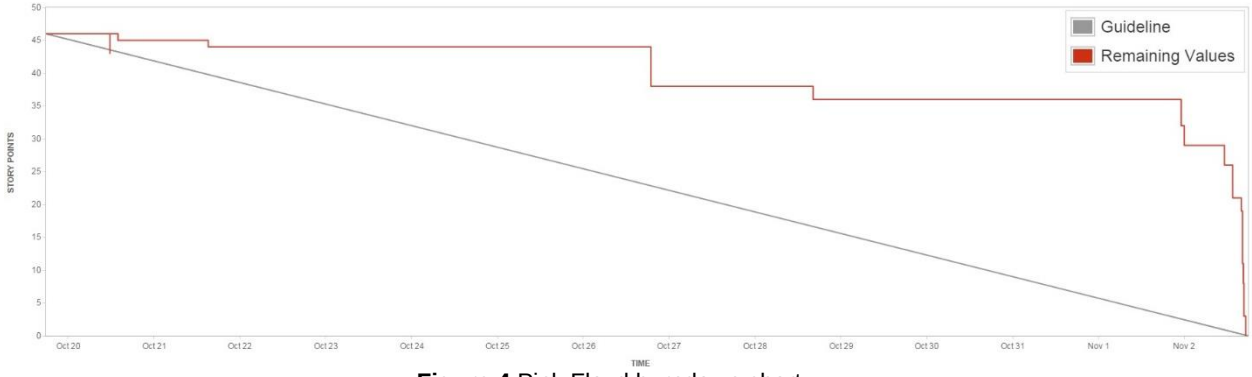

Figure 4 Pink Floyd burndown chart

Completed Issues

Key	Summary	Issue Type	Priority	Status	Story Points (46)
VNET-15	Node Config Tools - Test Config Definition	Story	Major	DONE	3
VNET-21	Node Config Core - PASSWORD type	Story	Major	DONE	3
VNET-22	Node Client Core API Error Reporting	Story	Major	DONE	2
VNET-24	Node Config Groups	Story	Major	DONE	3
VNET-25	Move Node Config Utils Class to utils package and protect utils	Story	Minor	DONE	1
VNET-26	Develop web page ver 0.2	Story	Minor	DONE	2
VNET-27	Node Config Core - FILE type serialization and deserialization	Story	Major	DONE	1
VNET-28	Application Server - Basic Requirements Definition	Story	Major	DONE	1
VNET-29	Node Config Core - OPTIONS type	Story	Major	DONE	2
VNET-30	Database Server - Basic Requirements Definition	Story	Major	DONE	3
VNET-31	Name Server - Basic Requirements Definition	Story	Major	DONE	3
VNET-32	OpenVPN Server - Basic Requirements Definition	Story	Major	DONE	3
VNET-33	Web Server - Basic Requirements Definition	Story	Major	DONE	5
VNET-34	Node Config Core - BYTESIZE type	Story	Major	DONE	2
VNET-35	Create mailbox of VirtNet team	Task	Minor	DONE	1
VNET-37	Management Core Architecture Design	Story	Major	DONE	8
VNET-38	Document Config Definition Format	Task	Minor	DONE	3

Figure 5 Pink Floyd issue list

5.2.3 Nazov Stavby

Number of sprint: 3

Name of sprint: Nazov Stavby

Sprint date: 03/Nov/15 4:16 PM - 09/Nov/15 2:58 PM

Number of issues: 12

Number of non-completed issues: 2

Number of story points: 44

Number of non-completed story points: 12

The third sprint named Nazov Stavby was primarily aimed at creation and setup of development environment, which is OpenStack platform. Also Debian Jessie image for OpenStack was created, on which the node client services will be run. Created was XML builder, which will be used for communication between core and clients.

Within the node client core script for automatic startup was created. Node client methods for pushing configuration and controlling service were created. Also tests for

node client core were corrected. Documentation and webpage changes based on review from team leader were performed.

As is shown in Figure 6, the sprint was planned on less than week and amount of story points was larger than how it was in the second sprint Pink Floyd. This resulted in a failure to complete two of planned issues. Migration of ngnlab.eu machines was not done due to extensive differences between source and destination environment and lack of time. The issue has been revised and will be planned to the next sprint. In addition, decision was made to create epics for more challenging issues. In other words, the issue will be distributed into more manageable issues. Entire list of completed and not completed issues is shown in Figure 7.

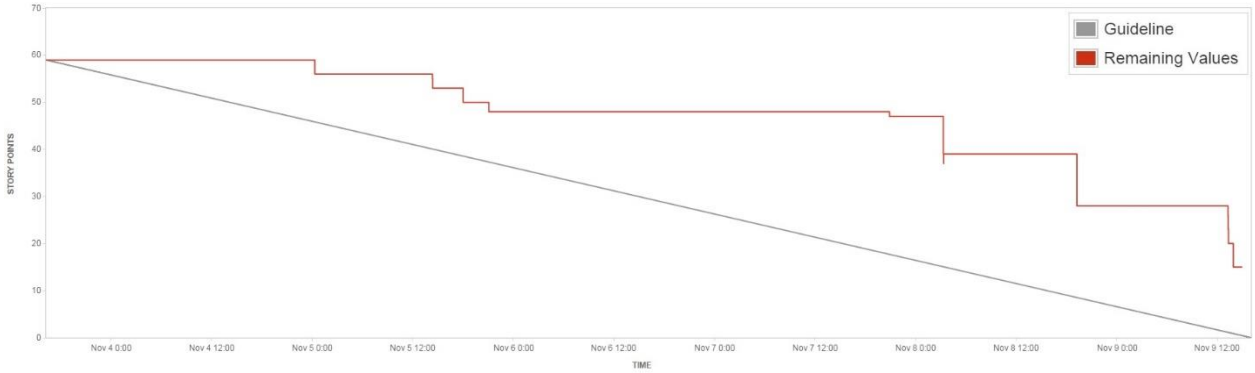

Figure 6 Nazov Stavby burndown chart

Completed Issues

Key	Summary	Issue Type	Priority	Status	Story Points (44)
VNET-7	Node Client Push Configuration API Method	Story	Major	DONE	3
VNET-8	Node Client Service Control API Methods	Story	Major	DONE	2
VNET-18	Node Client Automatic Startup	Story	Major	DONE	5
VNET-23	Node Client Core Testing Module Missing	Bug	Minor	DONE	2
VNET-36	Create Debian 8 cloud image	Task	Major	DONE	8
VNET-40	Node Config Core - Field Required Config Definition Field	Story	Major	DONE	3
VNET-41	Node Config Definition Test Ignores Duplicate Field Names In Group	Bug	Major	DONE	2
VNET-42	Node Configuration Parser should detect default value type mismatch	Bug	Major	DONE	1
VNET-43	Node Configuration Definition Format Version	Story	Major	DONE	1
VNET-44	Node Config Core - Field Default Value must be optional	Story	Major	DONE	3
VNET-46	Node Config Core - XML configuration builder	Story	Major	DONE	3
VNET-48	Configure OpenStack Network and Controller Nodes (single machine, currently ngnlab-router)	Story	Major	DONE	5
VNET-49	Configure OS compute server 1 (Cisco, 64GB RAM)	Story	Major	DONE	3
VNET-59	Translate team web page to english	Task	Minor	DONE	3

Issues Not Completed

Key	Summary	Issue Type	Priority	Status	Story Points (15)
VNET-53	Migrate ngnlab.eu machines to OS compute server 1	Story	Major	TO DO	13
VNET-54	Analyze options for migration of Xen machines to OpenStack	Story	Major	TO DO	2

Figure 7 Nazov Stavby issue list

5.2.4 Kratky Proces

Number of sprint: 4

Name of sprint: Kratky proces

Sprint date: 09/Nov/15 4:45 PM - 18/Nov/15 7:33 PM

Number of issues: 25

Number of story points: 88

The fourth sprint named Kratky Proces was primarily aimed at creation of documentation for 1st checkpoint. The documentation was divided into two main parts:

- Management documentation
- Technical documentation

Management documentation contains big picture, standards used in team, goals for winter semester, winter semester retrospective, project team meeting reports and sprint summaries.

Technical documentation contains system overview, project proposal, goals for winter semester and system modules. System modules part is divided into analysis, design, implementation and testing.

Management system bundles were designed in the concrete configuration and database bundle. Also the configuration bundle were implemented and reviewed. Environment and networking for NetCell development and testing was setup and VPN remote access to network was granted. Node configuration tools were upgraded.

As is shown in

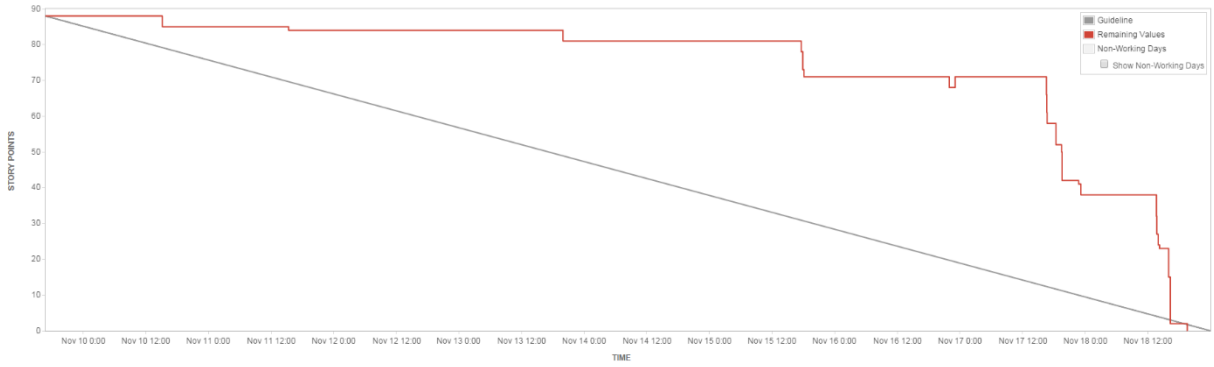

Figure 8, the sprint was planned on two weeks and amount of story points was significantly higher than in previous sprint, which resulted into a bad time management of team members. So main part of issues was resolved at the end of the sprint. Entire list of completed issues is shown in

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (88)
VNET-54	Analyze options for migration of Xen machines to OpenStack	Story	Major	DONE	8
VNET-56	Config Bundle SPI Development	Story	Major	DONE	3
VNET-58	Config Bundle Implementation Development	Story	Major	DONE	5
VNET-60	Node Configuration Tools - Generate config from live host	Story	Major	DONE	3
VNET-61	Node Configuration Tools - Push config to live host	Story	Major	DONE	3
VNET-62	Node Configuration Tools - Control remote service on live host	Story	Major	DONE	2
VNET-63	Create test for BYTESIZE field type	Story	Major	DONE	1
VNET-65	Setup Horizon in VM on oscompute-1	Story	Major	DONE	3
VNET-67	Management Document - Project Introduction	Task	Major	DONE	2
VNET-68	Management Document - Project Roles	Task	Major	DONE	1
VNET-69	Management Document - Application of Management Processes	Task	Major	DONE	5
VNET-70	Management Document - Sprint Summarizations	Task	Major	DONE	3
VNET-71	Management Document - Project Methodics	Task	Major	DONE	13
VNET-72	Management Document - Winter Semester Global Retrospective	Task	Major	DONE	3
VNET-73	Management Document - Used Methodics	Task	Major	DONE	2
VNET-74	Management Document - Task Export	Task	Major	DONE	1
VNET-75	Create remote VPN access to network	Story	Major	DONE	5
VNET-76	Setup environment and networking for NetCell development	Story	Major	DONE	3
VNET-77	Setup environment and networking for NetCell testing	Story	Major	DONE	3
VNET-78	Technical Document - Introduction	Task	Major	DONE	3
VNET-79	Technical Document - Global Goals For Winter Semester	Task	Major	DONE	3
VNET-80	Technical Document - Overall View of System	Task	Major	DONE	2
VNET-81	Technical Document - System Modules	Task	Major	DONE	5
VNET-101	Design Database Bundle	Task	Major	DONE	3
VNET-102	Discuss Config Bundle Structure and Methods	Task	Major	DONE	3

Figure 9Figure 7.

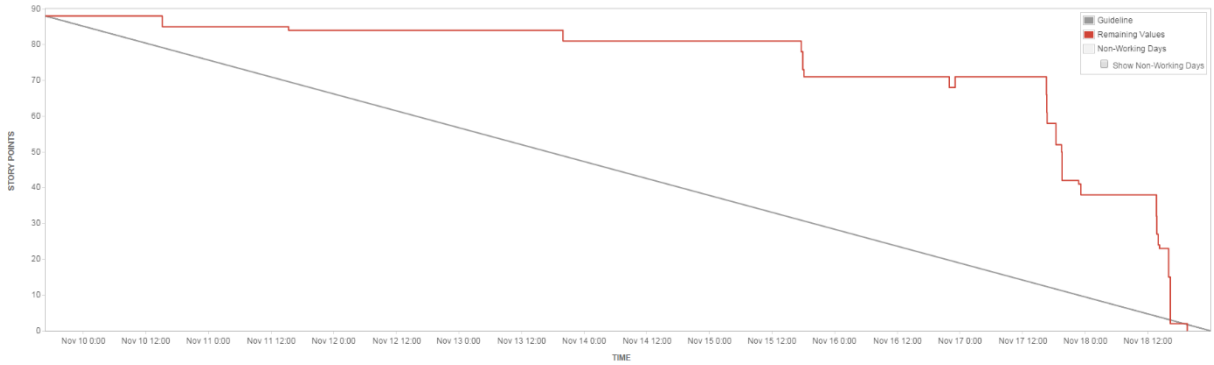

Figure 8 Kratky proces burndown chart

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (88)
VNET-54	Analyze options for migration of Xen machines to OpenStack	Story	Major	DONE	8
VNET-56	Config Bundle SPI Development	Story	Major	DONE	3
VNET-58	Config Bundle Implementation Development	Story	Major	DONE	5
VNET-60	Node Configuration Tools - Generate config from live host	Story	Major	DONE	3
VNET-61	Node Configuration Tools - Push config to live host	Story	Major	DONE	3
VNET-62	Node Configuration Tools - Control remote service on live host	Story	Major	DONE	2
VNET-63	Create test for BYTESIZE field type	Story	Major	DONE	1
VNET-65	Setup Horizon in VM on oscompute-1	Story	Major	DONE	3
VNET-67	Management Document - Project Introduction	Task	Major	DONE	2
VNET-68	Management Document - Project Roles	Task	Major	DONE	1
VNET-69	Management Document - Application of Management Processes	Task	Major	DONE	5
VNET-70	Management Document - Sprint Summarizations	Task	Major	DONE	3
VNET-71	Management Document - Project Methodics	Task	Major	DONE	13
VNET-72	Management Document - Winter Semester Global Retrospective	Task	Major	DONE	3
VNET-73	Management Document - Used Methodics	Task	Major	DONE	2
VNET-74	Management Document - Task Export	Task	Major	DONE	1
VNET-75	Create remote VPN access to network	Story	Major	DONE	5
VNET-76	Setup environment and networking for NetCell development	Story	Major	DONE	3
VNET-77	Setup environment and networking for NetCell testing	Story	Major	DONE	3
VNET-78	Technical Document - Introduction	Task	Major	DONE	3
VNET-79	Technical Document - Global Goals For Winter Semester	Task	Major	DONE	3
VNET-80	Technical Document - Overall View of System	Task	Major	DONE	2
VNET-81	Technical Document - System Modules	Task	Major	DONE	5
VNET-101	Design Database Bundle	Task	Major	DONE	3
VNET-102	Discuss Config Bundle Structure and Methods	Task	Major	DONE	3

Figure 9 Kratky proces issue list

5.2.5 Deladap

Number of sprint: 5

Name of sprint: Deladap

Sprint date: 23/Nov/15 3:49 PM - 08/Dec/15 10:54 PM

Number of issues: 24

Number of story points: 82

The fifth sprint named Deladap was primarily aimed at setup of virtual machines for node modules. Also development environment for all of the node modules was setup. Basic methods for getting basic node type information, node configuration definition and controlling node module services was implemented.

Bundle design continued in event logger and web server SPI. Also web server and database bundles were implemented. Test for configuration bundle were created. PAX Exam framework for testing in MS project was setup. Node configuration tools were design for OS windows.

As is shown in

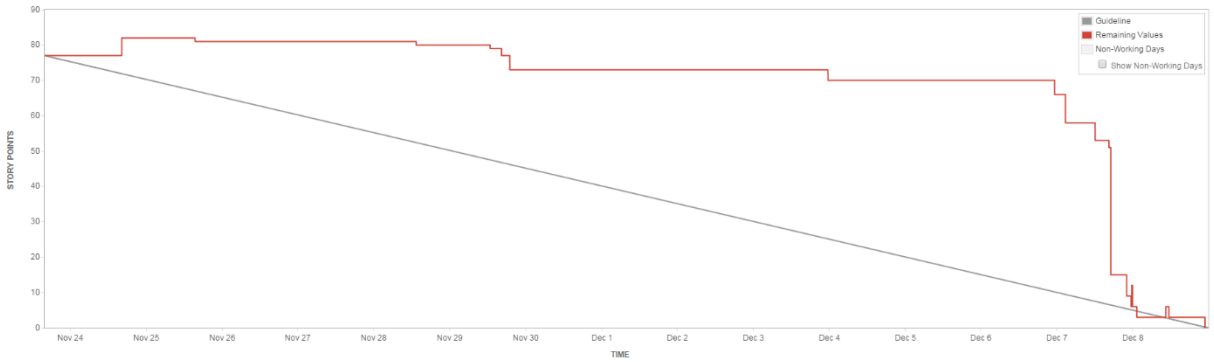

Figure 10Figure 6, the sprint was planned on less than week and amount of story points was 82, which resulted into a sprint extending for two more days. The sprint extending cause issue review inconsistency. Some of issue were accepted although they were incorrectly implemented. Also some issue were added after sprint started. Entire list of completed and not completed issues is shown in

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (82)
VNET-64	Change paths in Node Client Core	Bug	Major	DONE	1
VNET-104	Create License Headers in NodeClient Core	Task	Major	DONE	1
VNET-105	Node Client Error Reporting is ambiguous	Bug	Major	DONE	2
VNET-108	Node Client Core should pass root element to Node Client Implementation	Task	Major	DONE	1
VNET-109	Setup Junit4OSGi for MS project.	Task	Major	DONE	8
VNET-110	Create tests for Configuraiton Bundle	Task	Major	DONE	2
VNET-111	Node Config Tools should contain .bat executable for Windows users	Story	Major	DONE	1
VNET-113	Setup development VM for Database Server Node Type	Task	Major	DONE	2
VNET-114	Setup development VM for Application Server Node Type	Task	Major	DONE	2
VNET-115	Setup development VM for Nameserver Node Type	Task	Major	DONE	2
VNET-116	Setup development VM for OpenVPN Node Type	Task	Major	DONE	2
VNET-117	Setup development VM for Web Server Node Type	Task	Major	DONE	2
VNET-118	Implement basic functionality in NT module for Database Server	Story	Major	DONE	3
VNET-119	Implement basic functionality in NT module for Application Server	Story	Major	DONE	3
VNET-120	Implement basic functionality in NT module for Nameserver	Story	Major	DONE	3
VNET-121	Implement basic functionality in NT module for OpenVPN Server	Story	Major	DONE	3
VNET-122	Event Logger Bundle Implementation Development	Story	Major	DONE	8
VNET-123	Event Logger Bundle SPI Development	Story	Major	DONE	2
VNET-124	Implement basic functionality in NT module for Web Server	Story	Major	DONE	3
VNET-125	Web Server Bundle Implementation Development	Story	Major	DONE	13
VNET-126	Web Server Bundle SPI Development	Story	Major	DONE	5
VNET-127	Design Event Logger Bundle	Story	Major	DONE	3
VNET-128	Design Web Server Bundle	Story	Major	DONE	5
VNET-129 *	Database Bundle Implementation	Story	Major	DONE	5

* Issue added to sprint after start time

Figure 11Figure 7.

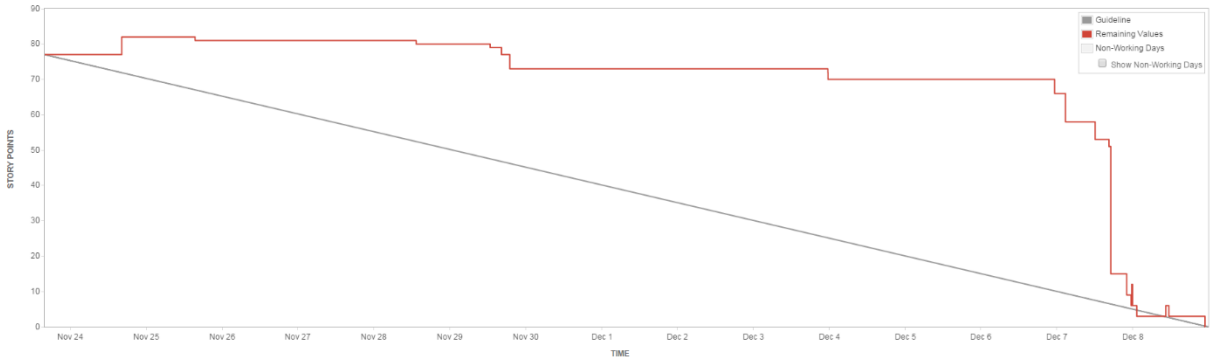

Figure 10 Deladap burndown chart

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (82)
VNET-64	Change paths in Node Client Core	Bug	Major	DONE	1
VNET-104	Create License Headers in NodeClient Core	Task	Major	DONE	1
VNET-105	Node Client Error Reporting is ambiguous	Bug	Major	DONE	2
VNET-108	Node Client Core should pass root element to Node Client Implementation	Task	Major	DONE	1
VNET-109	Setup Junit4OSGi for MS project.	Task	Major	DONE	8
VNET-110	Create tests for Configuraiton Bundle	Task	Major	DONE	2
VNET-111	Node Config Tools should contain .bat executable for Windows users	Story	Major	DONE	1
VNET-113	Setup development VM for Database Server Node Type	Task	Major	DONE	2
VNET-114	Setup development VM for Application Server Node Type	Task	Major	DONE	2
VNET-115	Setup development VM for Nameserver Node Type	Task	Major	DONE	2
VNET-116	Setup development VM for OpenVPN Node Type	Task	Major	DONE	2
VNET-117	Setup development VM for Web Server Node Type	Task	Major	DONE	2
VNET-118	Implement basic functionality in NT module for Database Server	Story	Major	DONE	3
VNET-119	Implement basic functionality in NT module for Application Server	Story	Major	DONE	3
VNET-120	Implement basic functionality in NT module for Nameserver	Story	Major	DONE	3
VNET-121	Implement basic functionality in NT module for OpenVPN Server	Story	Major	DONE	3
VNET-122	Event Logger Bundle Implementation Development	Story	Major	DONE	8
VNET-123	Event Logger Bundle SPI Development	Story	Major	DONE	2
VNET-124	Implement basic functionality in NT module for Web Server	Story	Major	DONE	3
VNET-125	Web Server Bundle Implementation Development	Story	Major	DONE	13
VNET-126	Web Server Bundle SPI Development	Story	Major	DONE	5
VNET-127	Design Event Logger Bundle	Story	Major	DONE	3
VNET-128	Design Web Server Bundle	Story	Major	DONE	5
VNET-129 *	Database Bundle Implementation	Story	Major	DONE	5

* Issue added to sprint after start time

Figure 11 Deladap issue list

5.2.6 Five Finger Death Punch

Number of sprint: 7

Name of sprint: Five Finger Death Punch

Sprint date: 01/Mar/16 10:00 AM - 16/Mar/16 4:45 PM

Number of issues: 29

Number of story points: 74

Number of non-completed issues: 28

Number of non-completed story points: 37

First sprint in summer semester was started in second week of the semester. First week of semester was used for discussion and analysis other opportunities. The whole team decided about two weeks sprint duration. This sprint contains main part of functionality of Netcell Node-Client Core, which is network service configuration. Parser for XML configurations were created in this sprint except of Database Server configuration and Web Server configuration. Web server configuration was already created but not reviewed, so we weren't able to close this issue.

Because of the illness of two team members, not all tasks were fully completed during sprint duration. Figure 14 displays all not-completed tasks and also removed issues from sprint.

As is shown in Figure 12, most tasks were completed in last days of sprint. This was caused by long code review. Most of the issues were done successfully which is shown in Figure 13.

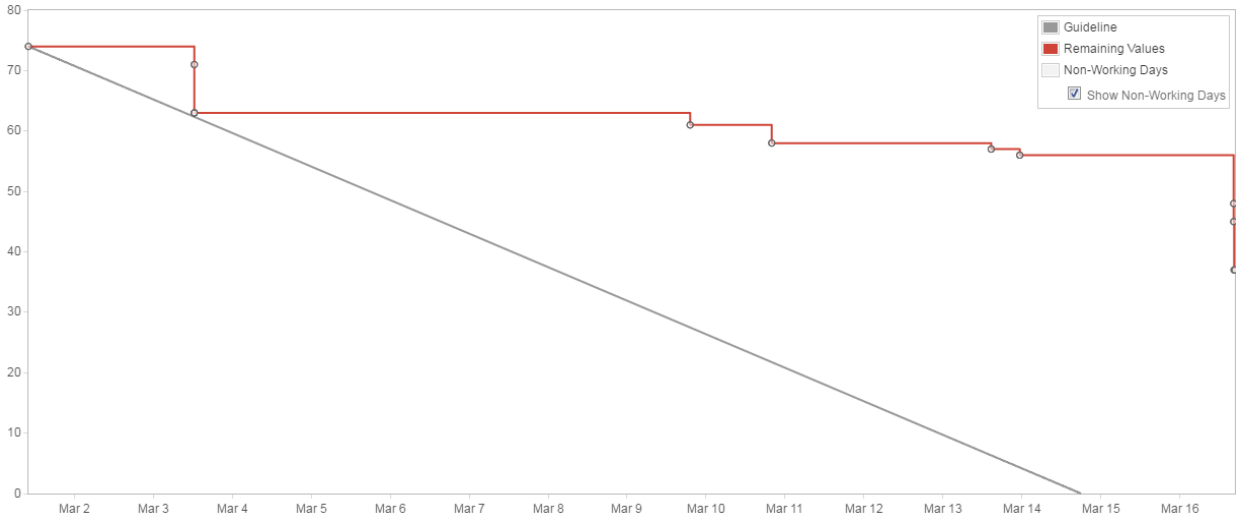

Figure 12: Five Finger Death Punch Burndown Chart.

Key	Summary	Issue Type	Priority	Status	Story Points (58)
VNET-130	Nodeclient-core, after kill main proces, child proces still exists. This is bug.	Bug	Major	DONE	3
VNET-132	Create Quality and Management of Processes Presentation	Story	Major	DONE	3
VNET-133	Create Documentation Management Presentation	Story	Major	DONE	3
VNET-134	Create Risk Management Presentation	Story	Major	DONE	3
VNET-135	Create Communication Management Presentation	Story	Major	DONE	3
VNET-136	Create Testing Management Presentation	Story	Major	DONE	3
VNET-137	Create Development Management Presentation	Story	Major	DONE	3
VNET-139	Merge and Submit Final Documentation to AIS	Task	Major	DONE	2
VNET-141	Technical Document WS Submit Modifications	Task	Major	DONE	8
VNET-142	Merge NetCell Components into Functional Prototype	Task	Major	DONE	2
VNET-154	Merge Management Documentation parts into one powerpoint presentation.	Task	Major	DONE	2
VNET-161	Database Bunble Service should not be created when DB connection was not successfull.	Bug	Major	DONE	2
VNET-164	Add dynamic to integration tester	Task	Major	DONE	1
VNET-172	Application Server Basic Config	Story	Major	DONE	1
VNET-173	VPN Server Basic Config	Story	Major	DONE	8
VNET-175	NameServer Basic Config	Story	Major	DONE	8
VNET-176	Identity Service SPI Development	Story	Major	DONE	3

Figure 13: Five Finger Death Punch completed issue list

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (37)
VNET-155	Improve logging in web services API servlet	Task	Major	TO DO	1
VNET-156	Ensure empty responses are not sent by web services API servlet	Task	Major	TO DO	2
VNET-157	Remove HelloApiRoute after debugging	Task	Major	TO DO	2
VNET-158	Portal support for web services	Story	Major	TO DO	3
VNET-171	Database Server Basic Config	Story	Major	IN PROGRESS	5
VNET-174	WebServer Basic Config	Story	Major	IN PROGRESS	8
VNET-177	Identity Service Implementation Development	Story	Major	IN PROGRESS	5
VNET-180	OpenStack Manager Service SPI Development	Story	Major	IN PROGRESS	3
VNET-181	OpenStack Manager Service Implementation Development	Story	Major	TO DO	8

Issues Removed From Sprint View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (23)
VNET-140	Management Document WS Submit Modifications	Task	Major	DONE	12
VNET-178	Portal Core Service Implementation Development	Story	Major	TO DO	8
VNET-179	Portal Core Service SPI Development	Story	Major	TO DO	3

Figure 14: Five Finger Death Punch not-completed issue list

5.2.7 DMS

Number of sprint: 8

Name of sprint: DMS

Sprint date: 22/Mar/16 10:00 PM - 11/Apr/16 5:37 PM

Number of issues: 15

Number of story points: 53

Second sprint of summer semester was mainly focused on functional test creation of Node Types Configuration developed in previous sprint. Uncompleted tasks from previous sprint were also added to this sprint and they were successfully completed in this sprint. Sprint also contained tasks focused on development of management core

which is responsible for management of developed Node Types Configurations. New services for provision of OpenStack functionality was added to management core in this sprint. As you can see on burndown chart shown in Figure 15, most of the tasks were completed right before the end of the sprint. Main reason of this issue was because task were hard to review because of the complexity of each task.

Figure 15: DMS burndown chart

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (48)
VNET-155	Improve logging in web services API servlet	Task	Major	DONE	1
VNET-157	Remove HelloApiRoute after debugging	Task	Major	DONE	2
VNET-171	Database Server Basic Config	Story	Major	DONE	5
VNET-174	WebServer Basic Config	Story	Major	DONE	8
VNET-177	Identity Service Implementation Development	Story	Major	DONE	5
VNET-180	OpenStack Manager Service SPI Development	Story	Major	DONE	3
VNET-181	OpenStack Manager Service Implementation Development	Story	Major	DONE	8
VNET-183	Nameserver - Implementation of Network Group into config	Story	Minor	DONE	3
VNET-184	Nameserver basic config functional tests	Story	Major	DONE	2
VNET-185	VPN Server basic config functional tests	Story	Major	DONE	2
VNET-186	Webserver basic functional tests	Story	Major	DONE	2
VNET-187	Application Server basic functional tests	Story	Major	DONE	2
VNET-188	Database Server basic funtional tests	Story	Major	DONE	2
VNET-189	Create post-install script for NodeClient services	Story	Critical	DONE	3

Figure 16: DMS completed issue list

5.2.8 Arctic Monkeys

Number of sprint: 9

Name of sprint: Arctic Monkeys

Sprint date: 11/Apr/16 6:02 PM - 26/Apr/16 12:44 AM

Number of issues: 9

Number of story points: 49

This sprint was the last sprint of summer semester. Sprint was focused on refactoring management server because of the OpenStack java OSGi libraries mismatch and Node Type Management Development. Based on product functionality final tests were created. User acceptance testing environment was created in OpenStack network and fist version of final product was deployed into this infrastructure. All tasks were successfully completed and as you can see on the burndown chart shown in Figure 17, tasks were finished and reviewed on regular basis.

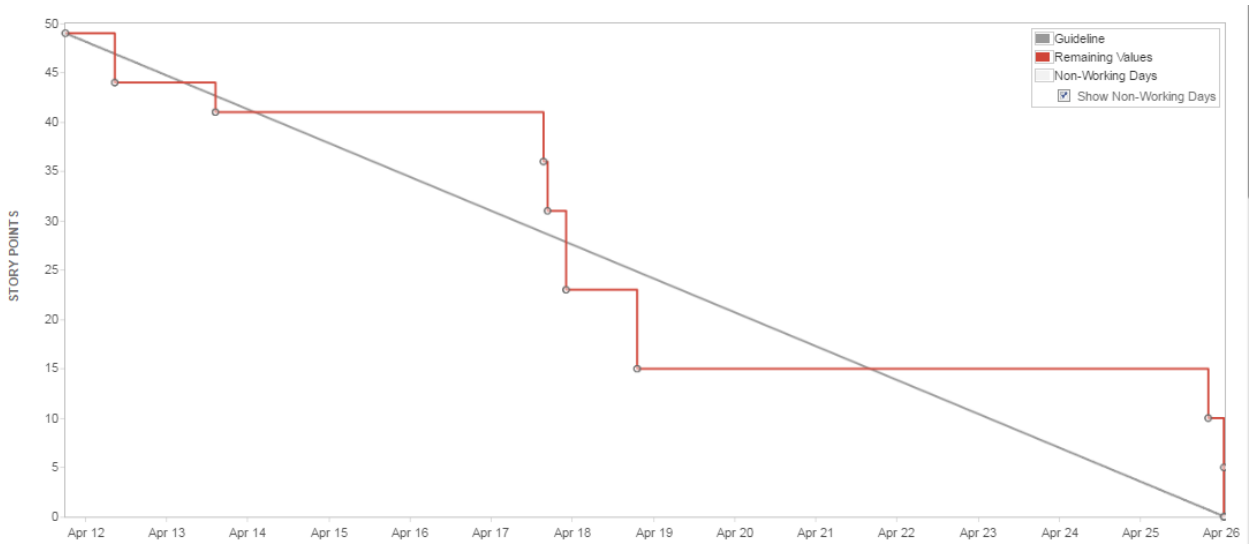

Figure 17: Arctic Monkeys burndown chart

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (49)
VNET-182	Netcell TaskManager Implementation	Story	Critical	DONE	5
VNET-195	Create Ethernet interface configuration Nodeclientcore	Story	Critical	DONE	3
VNET-196	Create and Add Debian Image to Openstack	Story	Major	DONE	5
VNET-197	Refactor Management Server OSGi Architecture to Maven Module Architecture	Story	Major	DONE	5
VNET-198	Netcell NodeType Manager Development	Story	Major	DONE	8
VNET-199	Netcell NodeType External Development	Story	Major	DONE	8
VNET-200	Create Testing Environment with all Netcell components.	Story	Major	DONE	5
VNET-201	Create API Client for Netcell Management Server.	Story	Major	DONE	5
VNET-202	Create Basic Test Cases for Testing Team	Story	Major	DONE	5

Figure 18: Arctic Monkeys issue list

6 Winter Semester Global Retrospective

This chapter is focused on how we fill goals from beginning of the semester. Our team created some goals for winter semester which are described in document Global Goals for Winter Semester, which is part of technical documentation.

As first decision made by team was choosing cloud platform for our product. We decided to use OpenStack. This decision was based on good knowledge of OpenStack modules and fact that OpenStack have very good documentation, which includes API calls for VNFs and network operations (create, delete, modify). Next completed task was design of our system. We decided to split system into two part nodeclient-core (which is serving VNFs) and nodeconfig-core (which is serving as management core).

Virtual network functions (services) were chosen on first team meeting and agreed by each team member. Basic functionalities of network services were specified in second sprint and will be updated continuously. Common virtual image of operating system was created and tested in OpenStack.

One of the hardest goals is setup development and test environment and migrate ngnlab which is located on university servers we used. Migration is technically difficult and time-consuming so we decided migrate this lab later, after more migration options will be found.

In addition to defined goals we made some other things which were not defined as main project objective. This includes whole methodic and technical documentation, web page of our team, logos and Git repositories. Also JIRA as issue tracker and Confluence as document storage setup was necessary for team work. All this mentioned things were done successfully, most of them in the beginning of the semester.

In the second half of semester we setup new development and test environment running on OpenStack and we successfully built VPN access to this environment. Unfortunately, migration of ngnlab was not yet done and probably will be not done at all because of the operating system versions incompatibility.

We also created development virtual machines for VNF services and uploaded nodeclient-core and implementation of service configuration to these instances. This consists of API, which allows user to configure network services stored in VM.

Compared to the original plan configuration of services is not completed yet. Implemented functionality allow user to start/stop network service, get status of network service or get configuration information about network service.

Second part was creation of database configuration, custom event logger setup, web server and web interface of Management core server. All mentioned functionalities were created except of web interface. All created functionalities do not cover whole functionality, other functions (like handling etc.) will be added in future development.

7 Summer Semester Global Retrospective

This chapter is mainly focused on how we continue in filling team project goals. Firstly, we summarized the done work in winter semester and we created plan for the summer semester. The plan describes goals, which we wanted to accomplish and work redistribution for each team member during the summer semester.

We also continue on developing VNF services, especially on implementation of uploading configuration to instances running these services. Functions which allows user to update configuration for network services stored in VM were created. Development of services providing defined virtual network functions was completed as it was scheduled. Also post install script and tests were created for every service.

One of the hardest goals was development of NetCell management core, where technical issues stopped us from successful finishing. As reaction to these issues we refactored code of management core, which cost us a lot of time. In addition the webportal service was not developed as we planned. We agreed on giving up the developing the webportal in java language, instead we have chosen php and javascript languages. This portal provides communication with all stable and fully developed management core API endpoints.

These problems led to protraction of sprint and overall work on team project. Also we were forced to cancel one sprint due to lack of time. Technical and time management issues culminated into delaying of product development and releasing.

From the perspective of organization, we regularly met at team meeting at least once a week. On team meeting we were discussing work progress and occurred issues, and preparing requirements for sprints. In addition to official meetings we organized extra sessions dedicated to important bugs and blocker tasks.

Overall development functionality provides fully functional VNF clients, which are ready for implementation into operable OpenStack network. Developed management core system contains most of the core components like identity service, database service, OpenStack service or task management service, which are needed for fully operable virtual topology management system.