Lassi Vainola

Automated Hardware Repair Process

Metropolia University of Applied Sciences

Bachelor of Engineering

Electrical and Automation Engineering

Bachelor’s Thesis

29 September 2021

Abstract

Author:	Lassi Vainola
Title:	Automated Hardware Repair Process
Number of Pages:	23 pages
Date:	29 September 2021
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation
Instructors:	Marko Rajala, Cloud Platform Development Engineer
	Timo Tuominen, Senior Lecturer

This thesis concerns process improvement of Nokia’s Data Center hardware issue ticket creation process. The main goal is to study and find a way to simplify the process and make comprehensive documentation about the renewed process.

The thesis begins by describing the target operating environment and the details of the current process. After the current process is explained, the problem areas are analyzed. The thesis uses a well-known process development method, Theory of Constraints. An overview to the used tools is given in the technical baseline.

The two main problems of the current process were examined together with the Nokia cloud expert and a solution for the problems was planned. The existing problems are interrelated, and this study can solve both of them.

In the practical phase the production plan was agreed upon, the automation software code was created, and the required variables and scheduling were defined. The renewed automated system was taken into production after a testing period. Feedback was collected from the support team and from the data center managers. There was one change required from the feedback and that was implemented into the software code.

The outcome of this thesis work consists of the automated ticket creation process description, software code changes and parameter settings as well as step-by-step instructions for the users. The main benefits gained from the improvements of this study are that the maintenance personnel will significantly save time because number of visits to the Data Center sites is reduced, and additionally solving time of the tickets will be faster due to the fact that several hardware issues can be covered with one ticket.

Keywords: Process, Hardware, Ticket, Scheduling, Data Center

Tiivistelmä

Tekijä:	Lassi Vainola
Otsikko:	Automatisoitu laitteiston korjausprosessi
Sivumäärä:	23 sivua
Aika:	29.9.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Automaatio
Ohjaajat:	Marko Rajala, Cloud Platform Development Engineer Timo Tuominen, Lehtori

Tämä insinöörityö käsittelee Nokian datakeskuksien laitteistoon liittyvien vikailmoitusten luontiprosessia. Työn päätavoitteena on tutkia olemassa olevaa prosessia ja löytää tapa yksinkertaistaa sitä ja luoda kattava dokumentaatio uudistetusta prosessista.

Insinöörityö alkaa kohdeympäristön kuvauksella ja nykyprosessin yksityiskohdilla. Nykyprosessin jälkeen analysoidaan ongelma-alueet. Insinöörityössä käytetään tunnettua prosessinkehitysmenetelmää, Theory of Constraintsia (pullonkaula-ajattelu), jonka jälkeen on yleiskatsaus käytetyistä työkaluista.

Nykyisen prosessin kahta pääongelmaa tarkasteltiin yhdessä Nokian pilviasiantuntijan kanssa ja ongelmille suunniteltiin ratkaisu. Olemassa olevat ongelmat liittyvät toisiinsa, tämä työ ratkaisee molemmat.

Opinnäytetyön toteutusvaiheessa sovittiin tuotantosuunnitelmasta, luotiin automaatio-ohjelman koodi ja pyydetyt muuttujat ja asetettiin prosessin ajoitus. Uudistettu ja automatisoitu järjestelmä otettiin käyttöön testivaiheen jälkeen. Palautetta kysyttiin datakeskuksien vastaavilta ja tukitiimiltä. Palautteena tuli yksi muutosehdotus, tämä sisällytettiin ohjelmistokoodiin.

Tämän insinöörityön lopputulos koostuu vikailmoitusten automatisoidun luontiprosessin kuvauksesta, ohjelmistokoodin muutoksista ja parametriasetuksista sekä yksityiskohtaisista ja vaiheittaisista ohjeista käyttäjille. Tämän työn parannuksista saadut tärkeimmät edut ovat, että huoltohenkilöstö säästää merkittävästi aikaa, koska vierailut datakeskuksiin vähenevät. Lisäksi vikailmoitusten ratkaisuaika nopeutuu, koska useista laitteisto-ongelmista saadaan tieto yhdellä vikailmoituksella.

Avainsanat: Prosessi, Laitteisto, Vikailmoitus, Aikataulutus, Datakeskus

Contents

List of Abbreviations

AWX:	Ansible WorX
GSM:	Global System for Mobile Communications
HPE:	Hewlett Packard Enterprise
iOS:	An operating system used in Apple mobile devices
IPAM:	Internet Protocol Address Management
NESC:	Nokia Engineering and Service Cloud – Organization
PSU:	Power Supply Unit
VLAN:	Virtual Local Area Networks
VRF:	Virtual Routing and Forwarding

1 Introduction

The purpose of this thesis is to describe one of Nokia’s processes related to hardware repair process in a cloud server environment. The aim was to make the process more streamlined and reduce the number of tickets generated for broken components. The main problems that are targeted to be solved are ticket handling time for broken hardware repair process, as well as reduction of visits that are required to the data centers, where the cloud servers are located. Reduction of visits is foreseen to be gained by limiting the amount of hardware repair tickets generated per week. This thesis project was carried out for Nokia Engineering and Service Cloud organization (NESC).

In chapter 2 the reader is introduced to Nokia’s history and one of its organizations in which the study is done. The target organization of this thesis (NESC) manages and develops the Nokia cloud server environment in several data centers globally. The organization consists of cloud engineers, developers, specialists, and architects. This chapter also introduces original process of the hardware issue handling and problem statement with goals of the study.

Chapter 3 introduces the planned process improvement method and explains the main tools and techniques used in this study. This section is intended to give a short introduction to each topic to provide the reader some basic background understanding about the technical environment, where the studied process is running.

The workflow of the hardware repair process automation is described in chapter 4, which consists of planning and scoping, implementation, testing, launching, and feedback collection. Planning and scoping were done in co-operation with the cloud platform expert to be able to find the best practice in the very early phase of the study. Implementation and testing were performed parallelly to find out the working solution by learning and realizing the issues immediately. Feedback collection was done after the system was taken into production.

The thesis end results are explained in chapter 5. This section lists the provided deliverables to fulfill the targets of the study. The analysis of the results for the hardware repair process development task as well as evaluation of the study are in chapter 6.

2 Environment and Background

This section gives an overview to Nokia and to the studied targeted operating environment within Nokia. As a starting point for the study, the current hardware issue handling process and the needed enhancements of the process are described. Finally, problem statements and goals of the study are outlined.

2.1 Nokia as a Company

Nokia is over 155-year-old company. From its beginnings in 1865 as a single paper mill operation, Nokia has found and nurtured success over the years in a range of industrial sectors including cable, paper products, rubber boots, tires, televisions, and mobile phones. [1.]

Nokia's transition to a primary focus on telecommunications began in the 1990s. The first GSM call was made 30 years ago in 1991 using Nokia equipment [1]. Rapid success in the mobile phone sector allowed Nokia to become by 1998 the best-selling mobile phone brand in the world. In 2011, to address increasing competition from iOS and Android operating systems, Nokia entered into a strategic partnership with Microsoft and finally in 2014 Nokia sold its mobile and devices division to Microsoft. [1.]

The creation of Nokia Networks, following the buy-out of joint-venture partner Siemens in 2013, laid the foundation for Nokia's transformation into primarily a mobile broadband network hardware and software provider. The acquisition of Franco-American telecommunications equipment provider Alcatel-Lucent greatly broadened the scope of Nokia's portfolio and customer base in 2015. Additional acquisitions have positioned Nokia to be a global technology leader in the communications industry. [1.]

Nokia's net sales in 2020 was 21,9 billion euros and it operates in roughly 130 countries. Nokia is known as a forerunner in technology development, with examples of 129 billion euros invested into research and development in the past two decades and 3500+ patent families declared as essential to 5G, as well as gained 9 Nobel prizes as a company. [1.]

2.2 Target Operating Environment of the Study

Nokia has four different business groups: Cloud & Network Services, Mobile Networks, Network Infrastructure and Nokia Technologies. Within Cloud & Network Services, the study is done for Nokia Mobile Networks - organization.

NESC Private Cloud delivers reliable and scalable commodity infrastructure as a service to all Nokia Business Groups, enabling them to focus on their applications and solutions rather than setup and management of the own underlying infrastructure. A view to a typical data center environment is given in figure below.

Figure 1. Nokia server racks in Espoo data center. [12.]

The operating environment for the study is at and about NESC data centers and their server hardware. NESC data centers are globally distributed to different regions: North America, Europe, and Asia.

Nokia data centers houses different manufacturers server racks, mostly they are from Hewlett Packard Enterprise (HPE) and Nokia’s self-designed Nokia Airframe Rackmount servers. The racks contain server nodes, storage nodes, and data and management switches.

Each data center has a spare part storage, from where replacements can be easily found. Mostly single point of failure is storage disks or power supply units where this study will focus on.

The issue ticket creation and handling process of the hardware issues are currently handled by the support team and corrected in the data centers by the maintenance personnel; it is described in section 2.3 Current process. The current process has certain identified inefficiencies and development needs, which are introduced in section 2.4 Problem statement.

2.3 Current Hardware Issue Handling Process

Currently the process of ticket creation of broken hardware is mostly manual for the support team. Figure 2 contains workflow of ticket creation for the support team.

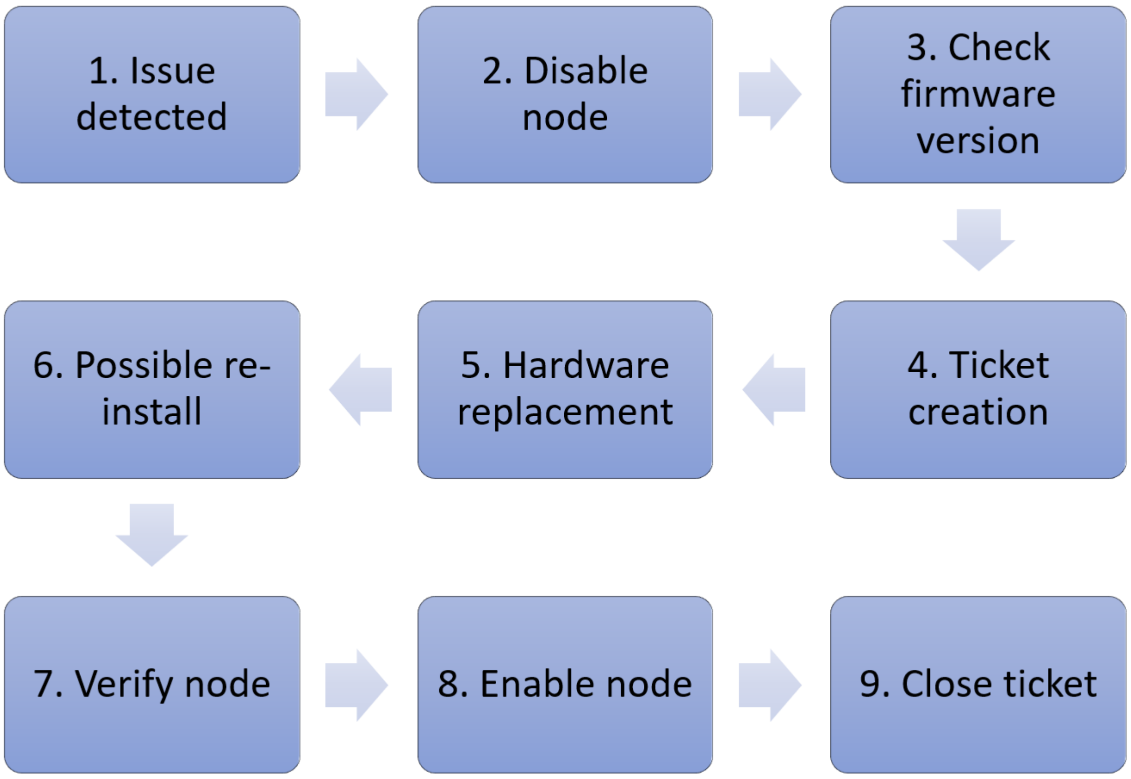


Figure 2. Current workflow of ticket creation.

Below is the ticket creation and handling workflow explained. The whole workflow is manual at this moment, except the automatic email which is sent to the support team in step 1.

1. Issue is detected by alert manager application and information of the broken node or parts is sent into the support team's email.

2. The support team ensures that the node/server is possible to be disabled from the cloud which allows node/server to be shut down. If not, then the support team moves the broken servers load to other node/servers and then disables the impacted node/server from the cloud.

3. The firmware of the node/server is updated by the support team to the latest version if it is not already running with the latest firmware version. In computing, firmware is a specific class of computer software that provides the low-level control for a device's specific hardware.

4. The support team creates a ticket into Jira. The ticket is then assigned to the corresponding regions data centers manager, who then assigns the job for the local maintenance personnel.

5. Local maintenance personnel go to the data center site and replaces the broken parts mentioned on the ticket in Jira. After replacing the broken parts, local maintenance personnel comments on the ticket for the support team what parts were replaced.

6. The support team checks if everything is correctly configured in NetBox, clears server cache (temporary memory) and if needed for a clean operating system, reinstalls operating system on the node.

7. The support team performs quality assurance script to help placing instances directly to the fixed broken nodes to verify the node/server functionality.

8. The support team enables the node back online.

9. The support team resolves the ticket in Jira. If an issue persists or the problem is not totally fixed, a new ticket is created.

2.4 Problem Statement and Goals of the Study

In this study, the aim is to enhance the current hardware issue handling process due to the following identified problems in the current process.

Problem 1: Slow and complicated process of hardware issue handling

Current process is too complicated because of many, and mostly manual tasks involved, as listed in chapter 2.3. Manual tasks affect slowness for the whole process.

Problem 2: Too many generated hardware issue tickets

Local maintenance teams of the data centers get too many hardware issue tickets, and currently one ticket is only for one node/server. The target is to collect such simpler hardware issues, which can be swapped without disabling the node/server from the cloud, in a one larger collection of alerts into a ticket that gets sent weekly. This would increase the time spent with one ticket, but the maintenance would only have to visit the site once a week for these types of issues. This lessens the time used on travelling to and from the data center and reduces travelling costs.

In this study, the goals are to streamline and fasten the existing hardware issue handling process, as well as reduce the number of generated tickets. The expected outcome from this thesis work is a simplified and automated hardware issue handling process for the organization.

3 Theory Baseline and Used Method

As described in the previous section, the problems of the current hardware issue handling process are clearly understood, and the goals of the thesis work have been set in advance by the ordering organization. Because of the pre-set targets and well-known development topics, from the process development methodology viewpoint Theory of Constraints has been selected among a wide range of process development methods. Theory of Constraints, and technical baseline to be able to make the necessary automations from the viewpoint of different tools used in the process, are explained in this section.

3.1 Theory of Constraints

A constraint or a bottleneck is a limiting factor which causes inefficiency in a process [9]. Generally, inefficiency typically leads either additional process cost or lost revenue. To overcome the constraint of bottleneck in a process, Theory of Constraints process development method aims to identify and remove the main bottleneck or a constraint from a process [9].

According to [9], Theory of Constraints consist of five steps that are described below in figure 3.

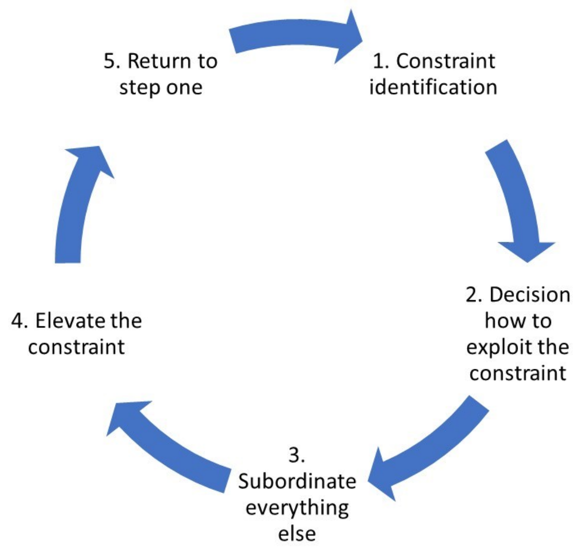

1. Constraint identification
2. Decision how to exploit the constraint
3. Subordinate everything else
4. Elevate the constraint
5. Return to step one

Figure 3. Steps of Theory of Constraints.

As part of this study, the steps are utilized as follows to optimize the automated hardware issue handling process:

1. Identify the system constraint

   Main system constraint has been identified to be the single fault indications, resulting frequently need to travel to the data center site

2. Decide how to exploit the constraint

   It has been decided to combine the fault corrections and thus limiting site visit amount to once a week per data center site

3. Subordinate everything else

   At the moment, the system is seen to function otherwise properly, assuming the planned optimization in step 2 is in place

4. Elevate the constraint

   No additional major modifications to the process are foreseen to be needed, thus step 4 can be omitted

5. Return to step one

   Identification of other potential constraints are for further study and not part of this thesis

3.2 Technical Baseline

This section gives an overview to main tools and techniques that are relevant for this study. The purpose of these sections is informational to understand better the actual process under development, instead of providing full introduction to each.

3.2.1 Ansible

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates. [7.]

Ansible’s main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans, even those not familiar with the program. [7.]

Playbooks work as an instruction manual for Ansible. At a basic level, playbooks are used to manage configurations of and deployments to remote PCs which is mostly how Ansible is used in this study. Playbooks are made to be read by people and are developed in a basic text language. [2.]

3.2.2 Prometheus

Prometheus is the software used in the alerts. It is an open-source systems monitoring and alerting toolkit. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open-source project and maintained independently of any company. [3.]

Prometheus can collect metrics from an application and infrastructure. Metrics are small concise descriptions of an event: date, time, and a descriptive value. Rather than gathering a great deal of data about one thing, Prometheus uses the approach of gathering a little bit of data about many things to help to understand the state and trajectory of the system. It has many features for monitoring metrics and providing alerts that can automate responses to changing conditions. [10.]

3.2.3 Ansible WorX

The AWX Project is a community driven open-source project sponsored by Red Hat, that enables users to better control their Ansible project use in IT environments. [4.]

AWX is built to run on top of the Ansible project, enhancing the already powerful automation engine. AWX adds a web-based user interface, job scheduling, inventory management, reporting, workflow automation, credential sharing, and tooling to enable delegation. [5.]

3.2.4 NetBox

NetBox is an infrastructure resource modeling application designed to empower network automation [6]. It is an open-source web application designed to help manage and document computer networks [11]. It encompasses the following aspects of network management:

- Internet Protocol Address Management (IPAM) - IP networks and addresses, Virtual Routing and Forwarding (VRF), and Virtual Local Area Networks (VLAN)
- Equipment racks - Organized by group and site
- Devices - Types of devices and where they are installed
- Connections - Network, console, and power connections among devices
- Virtualization - Virtual machines and clusters
- Data circuits - Long-haul communications circuits and providers
- Secrets - Encrypted storage of sensitive credentials

NetBox gives a user interface from the point of view of a network organization to help document IP addressing, while keeping the primary emphasis on network devices, system infrastructure, and virtual machines. [11.]

3.2.5 Jira

Jira is a tool developed by Australian Company Atlassian. This software is used for bug tracking, issue tracking, and project management. The basic use of Jira tool is to track issue and bugs related to software and Mobile applications. It is also used for project management. The Jira dashboard consists of many useful functions and features which make handling of issues easy. [8.]

4 Hardware Repair Process Automation

The expected outcome of the hardware repair process automation development was agreed to consist of an updated automated hardware repair process description, the actual script and files of the process and instructions for end users, for example data center managers and support personnel.

From the practical viewpoint, the work was split into phases that are summarized in the figure 4 below. The different steps are described in this section.

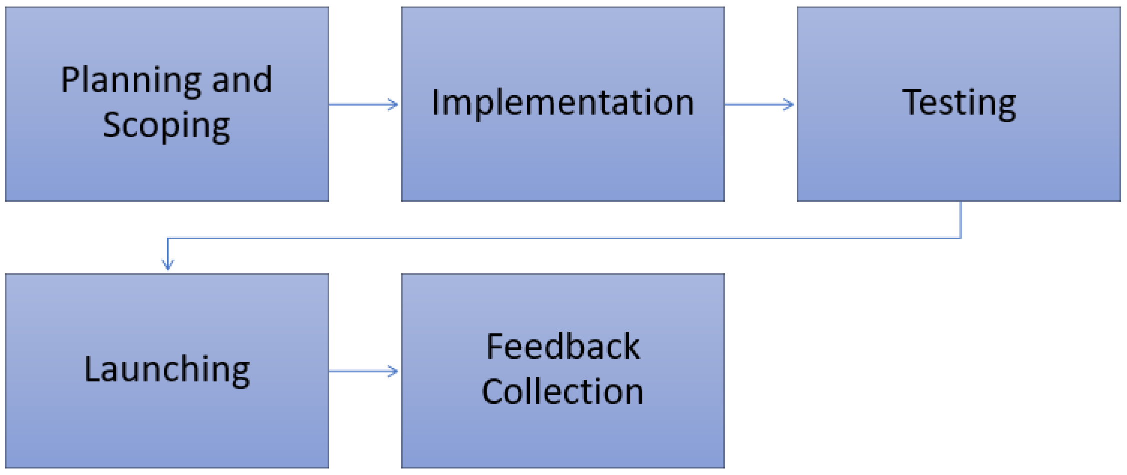


Figure 4. Work phasing in the thesis.

4.1 Planning and Scoping

First, meetings were arranged within a team that consisted of the manager and cloud expert in addition to the thesis worker. Discussions were had on what direction to move forward. Weekly meetings were scheduled for the three members. Then to collect the information on how the process currently works, the support team was contacted and got a hold of their workflow which contains details for the Jira tickets but also ticket creation in other systems. The focus is only on the Jira tickets. There was already a proof of concept available as baseline, which had been created by the Nokia cloud expert earlier.

To understand the data center environment, a visit was arranged to Telia data center in Helsinki, Pitäjänmäki. During the visit, a tour within data center was guided by a Cloud Architect who introduced the servers power supply, cooling, server layout and the server identification coding.

In another meeting discussion was on what program is going to be used to create the tickets. Python programming language is more versatile, but Ansible was chosen as it was easier to learn and read.

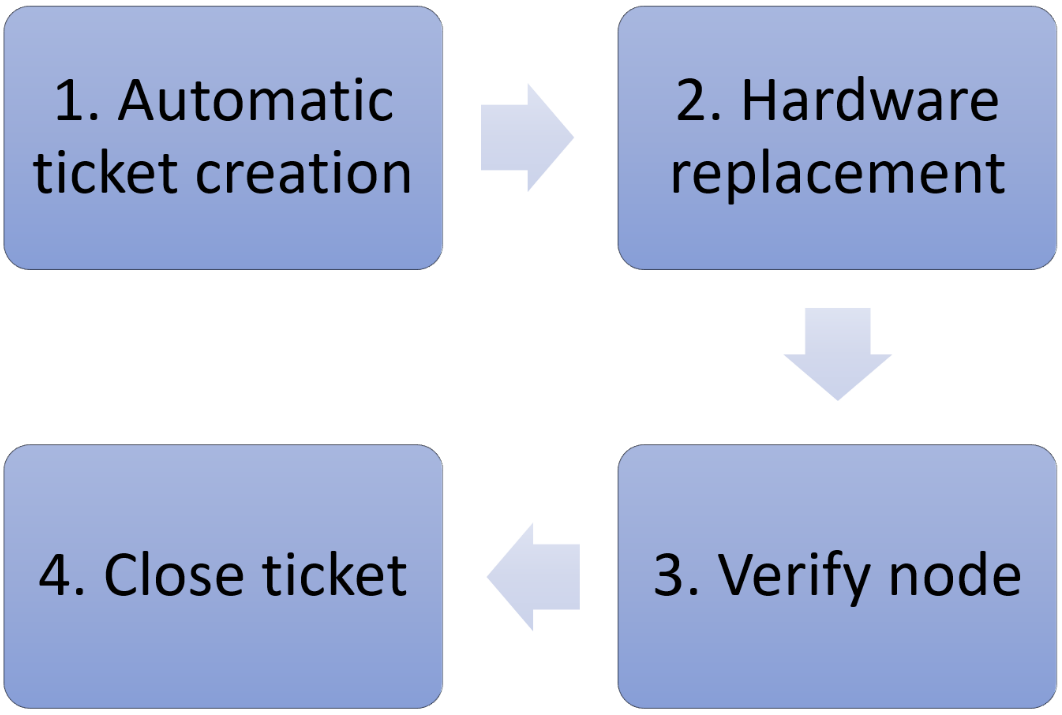


Figure 5. New workflow for ticket creation.

The new workflow for the ticket creation is explained below:

1. Ticket is automatically created for example every Monday at 7:55. Timing is adjustable in AWX scheduling by authorized users. Ticket contains an excel sheet which includes a list of servers that has broken components. These components are either storage drives or power supplies.

2. Data center manager sends the maintenance personnel to the site who will change the broken parts.

3. Maintenance personnel ensures the node is working properly.

4. Ticket is resolved and the maintenance personnel comments on the ticket, which parts were changed. Support team verifies this after the comments.

4.2 Implementation

The server list that will be attached to the ticket, which will be sent to the data center managers, is created at the same time as the Ansible playbook is ran. It is a task in the playbook which creates the server list using python and it gets the broken disk or power supply data from Prometheus. The Nokia expert had created the proof of concept for the server list creation and the python script which creates the excel sheet of servers with broken components.

The generated excel sheet contains a great amount of data about the broken components which makes it easier for the maintenance personnel to find the server and their broken parts. The detailed information that the excel sheet contains is:

- Node: Server name

- Sensor Name: Disk or Power Supply Unit name

- Event: Failure message

- Role: Server type (Openstack server or storage server)

- Node Type: Different variants for Openstack server and storage server.

- Device Type: Server brand and model

- Device Position: Server’s location in data center. For example, “in-bh-dc-z2-85-u40” (Two letter country code – two letter city code – site code – room name – server rack name – server location in rack)

The Ansible program requires a virtual environment to run. Nokia is using a system like AWX which works as a virtual PC where programs can be ran. The program and the needed files were deposited to Nokia GitHub where they can be loaded into the AWX. AWX has a scheduling option that can be used for scheduling the playbook to run once a week.

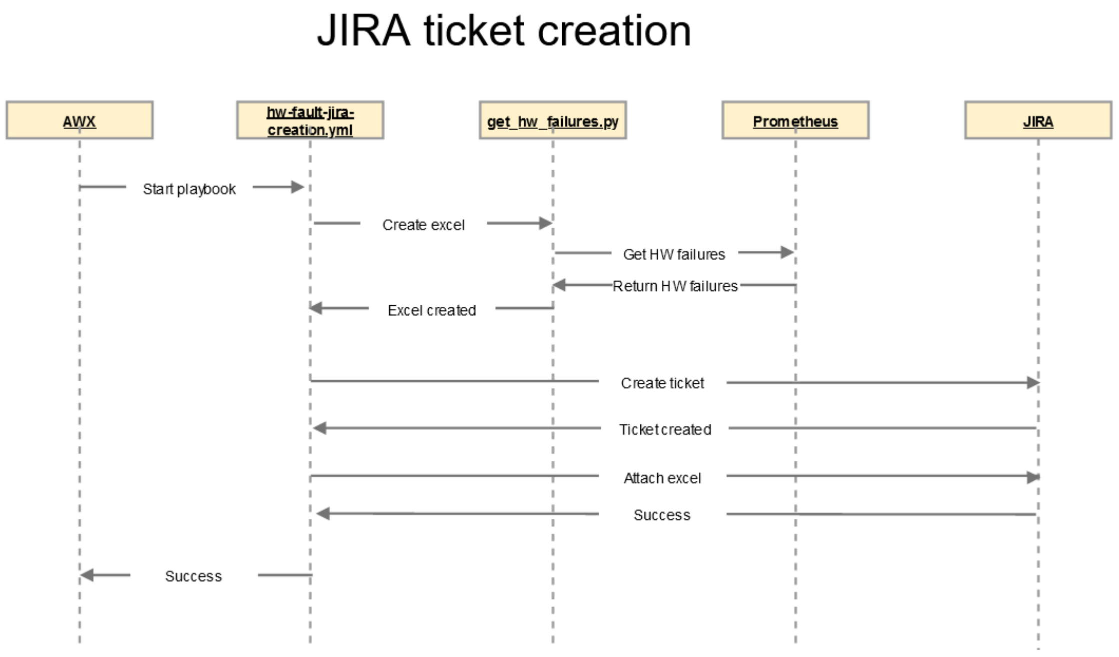


Figure 6. Jira ticket creation related software communications.

The software communications is explained below:

1. AWX starts the playbook “hw-fault-jira-creation.yml”

2. Playbook starts a python script “get_hw_failures.py” as its first task which extracts the list of servers with broken components from Prometheus into an excel file.

3. Playbooks second task creates the ticket into Jira with the needed extra variables used in the AWX schedule.

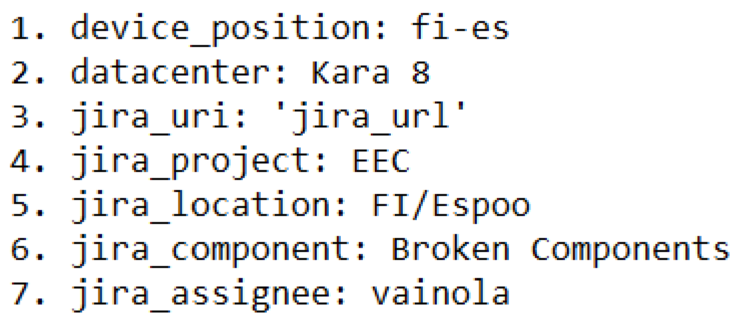
4. Playbooks third task is to insert the generated excel alert list to the Jira ticket as an attachment.

5. If all the variables were correct and the first ticket was created without error, AWX receives success information. If an error occurs, it is most probably in authentication, which means Jira is down or the functional entry user that is used in the program does not have access to Jira. The most likely reason is expired password or no access to Jira.

4.3 Testing

Testing was performed in a Nokia AWX test environment. The script was created in a Nokia GitHub test branch and after every modification made to the script, it was executed in AWX test environment. This was repeated until all the parameters were correct and the wanted result was achieved.

Running a job in AWX requires a template which has all the settings to use a correct project that fetches the playbook from GitHub. When the template parameters are correct, schedule can be created and executed. With schedule, job can be run for example every Monday at 7:55. Schedule also has extra variables that alter the playbooks original variables.



```
device_position: fi-es
datacenter: Kara 8
jira_uri: 'jira_url'
jira_project: EEC
jira_location: FI/Espoo
jira_component: Broken Components
jira_assignee: vainola
```

Figure 7. Example of extra variables.

Explanations of variables in figure 7 are given below:

1. Alert filter for metrics e.g., 'fi-es' filters servers from Espoo data centers.

2. Data center’s name which is set to the ticket's summary, title.

3. URL for used Jira.

4. Jira project name.

5. Location field for specific data center used in Jira.

6. Component field used in Jira. Broken Components is used in the hardware replacement process.

7. Jira username of who the ticket will be assigned to.

At the end of the testing phase the system was introduced to the end users and the feedback was requested. One feature was added after the feedback received, which was the location field for Jira tickets.

4.4 Launching

Automated hardware repair process was taken in use in September 2021. Test branch in GitHub was merged into the master branch which is used in the production side of Nokia AWX. Regional data center managers started to receive once a week the Jira tickets of faulty components.

As an outcome of this work the following items were provided:

1. Updated automated hardware repair process description.

2. Script and files of the process.

3. Instructions for end users. E.g., Data center managers and support personnel.

4.5 Post Launch Feedback Collection

After the launching an email was sent to the corresponding parties to collect further feedback about the process. The main questions were, if the process change has been helpful with the ticket handling and hardware replacement process and are there any ideas for developing it further. Additionally, more generic feedback was requested from the manager and the cloud expert about overall successfulness of the project.

5 Outcome of the Study

The goals of this study were to make the hardware repair process more streamlined and decrease the number of tickets created for each region data centers.

To fulfill the goals of the study, the following items were covered and provided:

1. Updated automated hardware repair process description is outlined in this document in section 4.1 figure 4.

2. Script and files of the process were transferred into the NESC GitHub. If there is a need to modify the script or files, this will be possible based on the existing versions.

3. Step-by-step instructions were created for end users, e.g. Data center managers and support personnel. Due to confidentiality requirements the instructions are provided for the specified user group in Nokia intranet.

6 Summary and Conclusions

The defined goal was to study the hardware repair process and its ticket management and make the ticket creation more streamlined by automating it and clustering small tasks into one bigger ticket per data center. The step-by-step instructions were simple for anyone if the schedule needs to be changed.

The first difficulties were getting all the accesses to Nokia’s systems and doing the whole study remotely from home, due to COVID-19. More difficulties were with learning to use all the tools and understanding how Ansible works, but it was close to other programming languages that were studied in Metropolia.

There was a visit arranged to the Telia data center in Helsinki, which helped to understand the structure of the data center and the components what will be changed using this process. Also, it gave practical knowledge to understand the whole process and the server position coding, which is a crucial part to identify the broken component.

For future developments this process could be made to be used with other ticket management systems than Jira, if Ansible has working modules for the other ticket systems.

References

1 Our history. 2021. Internet document. Nokia. https://www.nokia.com/about-us/company/our-history/. Visited 4.6.2021.

2 Working with playbooks. 2021. Internet document. Ansible. https://docs.ansible.com/ansible/latest/user_guide/playbooks.html. Visited 11.6.2021.

3 What is Prometheus? 2021. Internet document. Prometheus. https://prometheus.io/docs/introduction/overview/. Visited 30.6.2021.

4 The AWX Project. 2020. Internet document. Red Hat Ansible. https://www.ansible.com/products/awx-project/faq. Visited 27.7.2021.

5 The Inside Playbook, 5 Things You Can Do With AWX. 2017. Internet document. Red Hat Ansible. https://www.ansible.com/blog/5-things-you-can-do-with-awx. Visited 14.9.2021.

6 What is NetBox? 2021. Internet document. NetBox. https://netbox.readthedocs.io/en/stable/. Visited 14.9.2021.

7 About Ansible. 2021. Internet document. Ansible. https://docs.ansible.com/ansible/latest/index.html. Visited 14.9.2021.

8 What is JIRA? 2021. Internet document. Thomas Hamilton. https://www.guru99.com/jira-tutorial-a-complete-guide-for-beginners.html. Visited 14.9.2021

9 What is Theory of Constraints? 2006. Internet document. Lean Enterprise Institute. https://www.lean.org/common/display/?o=223. Visited 16.9.2021

10 What is Prometheus and Why Should You Use It? 2021. Internet document. Opsani. https://opsani.com/resources/what-is-prometheus-and-why-should-you-use-it/. Visited 22.9.2021

11 The Inside Playbook, Using NetBox for Ansible Source of Truth. 2020. Internet document. Red Hat Ansible. https://www.ansible.com/blog/using-netbox-for-ansible-source-of-truth. Visited 22.09.2021

12 Nokia. Nokia takes AirFrame data center solution to new performance levels with @intel's #XeonScalable processors. 2017. Twitter. https://twitter.com/nokia/status/884850990532497408. Visited 22.09.2021