**FORM A1**

**STUDENT PLAGIARISM DISCLAIMER FORM**

****

**PLAGIARISM DISCLAIMER**

STUDENT NAME: Raphael Salaja

STUDENT NUMBER: A00269349

PROGRAMME: Bachelor of Science (Honours) Software Design with Virtual Reality and Gaming

YEAR: 3

MODULE: Placement

LECTURER: John Barret

ASSIGNMENT TITLE: Final Report

DUE DATE: 6th May 2022

DATE SUBMITTED:

ADDITIONAL INFORMATION:

I understand that plagiarism is a serious academic offence, and that AIT deals with it according to the AIT Policy on Plagiarism.

I have read and understand the AIT Policy on Plagiarism and I agree to the requirements set out therein in relation to plagiarism and referencing. I confirm that I have referenced and acknowledged properly all sources used in preparation of this assignment. I understand that if I plagiarise, or if I assist others in doing so, that I will be subject to investigation as outlined in the AIT Policy on Plagiarism.

I understand and agree that plagiarism detection software may be used on my assignment. I declare that, except where appropriately referenced, this assignment is entirely my own work based on my personal study/or research. I further declare that I have not engaged the services of another to either assist in, or complete this assignment.

|  |  |
| --- | --- |
| **Signed:** | **Raphael Salaja** |
| **Date:** | **5th May 2022** |

Table of Contents

[Introduction 4](#_Toc102388736)

[Responsibilities and Duties 4](#_Toc102388737)

[Technical Achievements 6](#_Toc102388738)

[Challenges 10](#_Toc102388739)

[Skill Development 11](#_Toc102388740)

[Experience Retrospective 13](#_Toc102388741)

[Conclusion 15](#_Toc102388742)

**Table of Figures**

[Figure 1: Ericsson Test Campaign Page 5](#_Toc102388626)

[Figure 2: Diagram Showing The Architecture of CBRS 6](#_Toc102388627)

[Figure 3: Location of The deployments.yaml File 8](#_Toc102388628)

[Figure 4: Output of The New Command 8](#_Toc102388629)

[Figure 5: Snippet of The Confluence Spike Page 9](#_Toc102388630)

[Figure 6: A Story on Team's The Jira Page 11](#_Toc102388631)

[Figure 7: Analysis of Daily Log Sheets 13](#_Toc102388632)

# Table of Abbreviations

**Abbreviations Definition**

CBRS Citizens Broadband Radio Service

CBSD Citizens Broadband Service Devices

CLI Command Line Interface

CPI Certified Professional Installer

DC Domain Coordinator

ENM Ericsson Network Manager

LTE Long Term Evolution

PO Product Owner

SAS Spectrum Allocation Service

# Glossary

**Domain Coordinator** allows a domain name to keep certain information from appearing in public 6

**pair programme** two developers working together on one computer 4

**spike** a story that cannot be estimated until a development team runs an investigation 4

# Introduction

In this report, I will outline my experiences and I will reflect on my entire placement. I will go into detail about my responsibilities, describe my duties, my technical achievements, any challenges I faced, and I will finally reflect on my experience on my placement.

# Responsibilities and Duties

During my time at Ericsson, I was assigned to a newly formed team under the Citizens Broadband Radio Service (CBRS). This team was known as ‘Team America’ and is comprised of one product manager, one scrum master, three senior engineers, four junior engineers, and one co-op, which was me. Due to the team being newly formed, most work that is assigned to the team relates to documentation, training to understand the product, workshops, and investigations. For my first few weeks, a vast majority of my time was spent in training and workshops to understand the new tools and product that I and the team would be working on. I was also allowed to pair programme with the senior and junior engineers to monitor the work that they were doing, so that I would be able to view the work being done in real time. One example of this was a spike which consisted of setting up web proxy ports so that the environment can communicate correctly with them.

Subsequently, I was assigned to creating test campaigns for future spikes that would be undertaken. Test campaigns are used to manage a set of test cases over a specific period. These allow for the validation of the correct functionality for the software that will be developed. This was a new experience for me as I had never known of test campaigns prior to this. Figure 1 shows the page that is used to create test campaigns.

A screenshot of a computer

Description automatically generated

Figure 1: Ericsson Test Campaign Page

This opened my eyes to the fact that there are many tools within the industry that I should be able to learn and use. This is a key part of being a software developer, the ability to adapt and learn new technologies rapidly. I helped my team member complete these as a side task for a sprint period.

Another duty that I was given was that I had to help on a spike regarding the refactoring of a logging tool as it was returning too much information. This was my first experience of seeing and working on the product hands on. This task was assigned to me and two other junior engineers. The task needed to be completed because the information that the logger was returning contained sensitive data such as the user’s password and email address. This could lead to a security breach and the mishandling of an individual’s data. It was interesting to work on the tool and see the inner workings of the backend. It gave me a greater understanding of the product that the team was working on.

# Technical Achievements

In this section, I will give an insight into a major technical achievement that took place during my time spent at Ericsson. In my last sprint, I was assigned a spike to complete regarding the getting and storing of meta information in the Domain Coordinator (DC). I will give an overview of what the CBRS DC is so that I can explain the achievement in greater detail.

CBRS is a technology allowing the allocation of Long-Term Evolution (LTE) services in the Band 48 (B48), using a central Spectrum Allocation Service (SAS) to allocate local spectrum while avoiding interference with the incumbent users of the frequency band. The technology consists of the Citizens Broadband radio Service Devices (CBSD) which connects to a central Spectrum Allocation Server through a Domain Proxy. The Ericsson CBRS Domain Coordinator is a product providing a Domain Proxy and the management logic to convert Band 48 radio nodes into CBSDs. Ericsson CBRS Domain Coordinator is deployed on the Ericsson Network Manager (ENM) which allows it to take advantage of the deployment and reliability advantages of this platform. **Figure 2** below shows a diagram of the CBRS Arachitecture.

Diagram

Description automatically generated

****Figure 2:**** Diagram Showing the Architecture of CBRS

The main function of the Domain Coordinator is to obtain grants from SAS and configure the RAN cells to transmit on the authorized frequencies and use the authorized channel bandwidth to ensure the transmit power is within the authorized power. The architecture of the Ericsson CBRS Domain Coordinator is based on the ENM. ENM contains the Domain Coordinator itself which is a coordinator between states and notifications of the underlying RAN and the SAS. Configuration of the cells and collection of measurements is via existing ENM Netconf interface. An installer interface is provided in ENM to allow a Certified Professional Installer (CPI) to install and configure the Domain Coordinator and CBSDs in the system.

This spike that I was working on was formed from a larger task which involves getting information on which DC is running. On the install or the start-up of the DC we needed to get the meta-information using either Kubernetes, Helm, and or Bash scripts and then we needed to store this information into an environmental variable which could called in the Command Line Interface (CLI). This meant that we needed to make a new CBRS CLI command which would be added to get this information from the DC by using an already existing DCM REST interface that calls the DC REST interface. This information would then be displayed to the user in a digestible and understandable form.

This was needed because if the customer runs into a problem the customer can run this command, which will return all the information related to the DC. With this information we can then check for errors in compatibility, missing elements, outdated product versions, etc.

There are several ways we could have implemented this. We can use Helm commands, Kubernetes commands, or a script that passes it as a system property. We could also store them in an environmental or system variables. Another way could have been to store them in a static file. A key requirement was that all meta-information stories would have to work for all three types of deployment. ENM and the Cloud would take priority. A fallback may be considered later. Some of this information will be used internally and some reported to the user. For example, the availability zones probably would not be exposed. This information can be queried at start up and installation time and we had to keep in mind that this information should not change without the system being redeployed.

After multiple meetings with my teammates the course of action was to store the meta-information in environmental variable, then on the cloud we would pass the information to a pod if the Kubernetes environmental variable was required in the other deployments. We can use jobs-conf or standalone conf files to get the environmental variable where we can write bash in these files to help with logic, and then we can set them as a system variable that can be accessed by the code.

After this deliberation and conclusion, my first course of action was to find a command that would return the Kubernetes version. The one that I found was – **kubectl version**. This would return the version of Kubernetes that was installed along with the client and server version. With this information, I then had to find out where to put this command to be called. I faced some issues with this as I was unfamiliar with the layout and structure of the project. This meant that I needed to ask for help from my colleagues to guide me onto the right path. He informed me that we should put the command into the deployments.yaml file. The location of the file can be seen in Figure 3.

Text

Description automatically generated

Figure 3: Location of The deployments.yaml File

A deployment file is an object in Kubernetes that lets you manage a set of identical pods. This allows for the environment variables to be made and then for them to be called in the CLI. This meant that I had to use another command to retrieve the Kubernetes version as I couldn’t use the one found previously as that is dependent on bash scripts. The one that I found was – **capabilities.kubeversion**. This is a Helm command that returns the Kubernetes version installed. After testing my new additions in the application, I was able to return the Kubernetes version that was installed, which can be seen in Figure 4 below.

Graphical user interface, text, application, email

Description automatically generated

Figure 4: Output of The New Command

Following this, I had to create the documentation of my findings. This was stored on a Confluence page for the respective spike. This took some time as I had to explain in detail the steps and reasoning for each of my actions. Figure 5 shows a snapshot of the Confluence page that I worked on.

Graphical user interface, text, application

Description automatically generated

Figure 5: Snippet of the Confluence Spike Page

I had to book a meeting with my Product Owner (PO) to discuss the outcome of my spike. This is known as the review process. At first there were a lot of comments and changes that had to be made to the spike. This was because there was some missing information that was critical to the investigation. Due to my lack of experience, I did not know that these were needed. This led me to have consistent back and forth with my PO about the relevant issues. After some discussion and debate we were ablet to settle on a valid and full confluence page with all the necessary information about the spike. I was then able to close off the issue on the Jira page and hand in the spike which was a great achievement for me.

# Challenges

The challenges that I faced during my placement came in many shapes in forms. Some of the difficulties I encountered were mostly incidental in nature. For example, the primary issue I encountered when working on the spike was waiting for other team members to complete their spike, which was tied to ours. This was a blocking issue that prevented me and my team from completing a specific piece of work. We dealt with this by temporarily blocking off the spike due to the issue being out of our control.

Another issue I faced was adjusting to the new environment in which I would be working. This new environment is being developed mostly using virtual machines, with much of the work being done through the command line. I overcome this obstacle by asking my co-workers questions about how to do specific jobs, as well as asking them to walk me through the configuration of virtual machines and other software that I was unfamiliar with. This offered me a better knowledge of the new workplace, the tools I would be working with, and it allowed me to engage with and get to know my team mates better.

Understanding a spike related to the setting of a web proxy was a significant challenge for me. The specifics of the spike were that we should look at how to obtain cloud information on which Domain Coordinator standalone is running. This work presented some difficulties for me due to my lack of knowledge with cloud technology. However, this compelled me to persevere because it was a brand-new area of software in which I would be dabbling in. This, in turn, would provide me with more knowledge on the subject in the future. I discussed this dilemma with my teammates, and they walked me through the process of comprehending the systems.

Furthermore, in my most recent sprint, I encountered a problem with the use of.yaml files and associated systems. I was unfamiliar with working with.yaml files, so I had to conduct some web study to figure out how they worked. I ran into a problem that is frequent when working with these files: correct indentation. This Is due to the syntax of the language being indentation sensitive. After realizing how critical this was, I was able to rewrite my code and solve my difficulties.

# Skill Development

As a software developer, you must be able to adapt and evolve consistently with the rapid development of new technologies. When I started my placement, I was met with the same objective to learn and acquire new skills while still being able to complete and engage in the team sprints. In this section, I will explain and outline the new skills and technologies that I learned and acquired.

The first tool that I will talk about is Gerrit. Gerrit is a web-based team code collaboration tool that is free to use. A team of software engineers can use a Web browser to accept or reject each other's revisions to their source code. It is tightly integrated with Git, a distributed version control system. This technique was like git, but it placed a larger focus on code review and ensuring that the code on the master is complete and functional code.

Another technology that I had to learn was Kubernetes. Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project. This was a technology that I was highly unfamiliar with due to me not having any cloud experience. This did not stop me from learning about it in detail and with the help of my colleagues, I was able to grasp it.

Software developers usually engage in an Agile workflow. There are many tools that help accommodate this flow. One common tool used in large industries is Jira. Figure 6 below shows an example of a story on the Team’s Jira page.

Graphical user interface, text, application, email

Description automatically generated

Figure 6: A Story on Team's The Jira Page

Jira is part of a suite of solutions meant to assist teams of all sizes manage their workloads. Jira was initially intended to be a bug and problem tracker. But now, Jira has matured into a strong task management platform for many types of use cases, from requirements and test case management to agile software development. With this tool we were able to keep track of spikes and see which spikes needed more help or tasks that could be closed off and submitted.

The ability to operate in a team is something that will remain with me for a long time. Because most things are created as a team, and teamwork is essential in growth. My team was also incredibly kind and considerate. We quickly became friends, which made it easier for me to blend in. One thing I'll remember is to always ask for assistance. This is critical since it indicates that you want to get the solution quickly and don't want to squander the team's time, as time is the most valuable thing someone can have. However. when asking for assistance, I learned that you must show evidence of work before asking for help. You do not want to waste the time of your team members.

# Experience Retrospective

In this section, I will analyse my time spent in Ericsson in detail. I will do this by explaining each training area code that I partook in and outlining the type of activities that they entailed. After this I will go over how I will make use of this experience for my future endeavours. During my time on placement, I was tasked with reporting and documenting my working hours into daily logs. Figure 7 below shows a chart of my time spent.

Figure 7: Analysis of Daily Log Sheets

From this chart we can see that the most dominant training area is business requirements. This could be the case because the product that my team was under was still in early development, this meant that there was more learning, training and workshops that needed to be done for us to understand how to operate on the system. However, as the weeks progressed, we can see that that figure started to decrease, and architecture and development was higher. This was because the team started to take on more spikes that required coding and the team started to entrust me with more technical work. One downside of the experience however was the lack of testing I got to do, as we can see we did no testing at all. This could be down to the fact that the system required no testing and there was not enough code to warrant a testing regime. Now I will go into detail about what each of the Training area codes relate to and what work I did to achieve them on my placement.

In software there are many prerequisites that need to be in place before one can conduct technical development. Before a developer can engage in code, they must first understand what it is they are working on. This can involve many things such as participating in workshops corresponding to an area of work that needs to be learnt, training courses to improve necessary skills, meeting with the end user or customer to discuss their acceptance criteria for the product and many other things of that nature. During my placement I partook in these things and more team related activities such as team daily stand ups which are used to learn about the progress of every team member. This helps to outline any blockers or updates relating to the user’s experience for both the day before and the current day of work. Another requirement that is necessary is documentation, which I did a lot of during the end of my tenure. This is a crucial part of software as documentation can relay information to both the customer and anyone else who wants to work on the project.

Software is built on many structures, architectures and requires intensive designing to produce a product of great quality. Examples of architecture and design activities would be investigating and researching how to complete a task. I particularly enjoyed this training area as it gave me a greater understanding of how to plan out a task before executing it. I did this with my colleagues during the early stages of my placement and near the end of it.

Development is about getting hands on and doing work on the product itself. During the start of my placement, I was mainly pair programming, so I was not making changes to the code myself. This was due to my lack of exposure and experience with the product, so it was better for me to watch and learn how the code is structured. During the end of placement, I was tasked with my first solo spike. This is where I did the most development and this was one of the most interesting and engaging periods of my time.

Working code and a product that is meets the expectations of the customer requires a lot of testing. Testing is a key part of the software development cycle. Without it, code is likely to be faulty and fail. There are many forms of testing, the most common one would be to conduct unit tests. Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unfortunately, during my time here, I was not able to conduct any testing to this extent as the ode that we were working on was not of the scale that would warrant a full testing regime.

Getting feedback on a piece of work is necessary to achieve a product that meets the expectations of the customer and is of good quality. Feedback can also be used to determine where a team needs to improve. For example, after every sprint, the team would conduct a team retrospective to see where the team did well and where the team can improve. Another form of feedback is to engage in discussion with the customer of the product. This is done to figure out if the product being made is what they wanted. Delivery would relate to the closing of off a task either through a team demo or a review with the product manager.

Looking back on my time at Ericsson, I would say that it was a very helpful experience that brought me a lot experience, taught me new skills, and helped improve me as a software developer. Learning how to work in a team was a skill of mine that improved greatly after working here. Teamwork is a vital part of working in the software development industry, this is because cooperation on products leads to faster delivery times and products of greater quality. In the future, I will take all these experiences with me, and it will help me in my endeavours in the software industry.

# Conclusion

In this report, I outlined my experiences and I reflected on my entire placement. I described my responsibilities and duties, my technical achievements, any challenges I faced, and I will finally I reflected on my experience on my placement.