Development of Remote Diagnostic and Management Software

Project Report submitted for fulfilment of the
requirements for the Award of the Degree of

Master of Engineering
In
Electronic Instrumentation and Control

Submitted by
Mandeep Singh
801751002

Under the Guidance of

Host Mentor
Pankaj Joshi
Senior Staff Engineer
Stryker Corporation

Faculty Supervisor
Dr Moon Inder Singh
Assistant Professor
TIET

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

2019

Electrical and Instrumentation Engineering Department
Thapar Institute of Engineering & Technology, Patiala
(Declared as Deemed-to-be-University u/s 3 of the UGC Act., 1956)
Post Bag No. 32, Patiala – 147004
Punjab (India)

Table of Contents

List of Figures

List of Tables

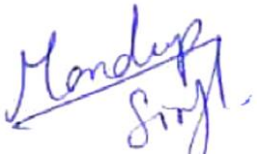
List of Abbreviations

Table 1 List of Abbreviations

Abbreviation	Explanation
TCP	Transmission Control Protocol
RDMS	Remote Diagnostic and Management Software
IoT	Internet of Things
OS	Operating System
SDLC	Software Development Life Cycle
NPD	New Product Development (team)
PKI	Public Key Infrastructure
POC	Proof of Concept
SRS	Software Requirement Specification
VB	Visual Basic
CLR	Common Language Runtime

Declaration

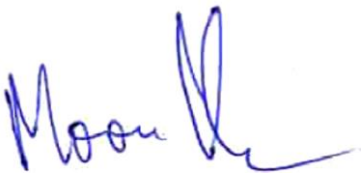
I hereby certify that the work which is presented in project report entitled, "**Development of Remote Diagnostic and Management Software**", is carried out in Stryker Corporation for the fulfilment of the requirements for the award of the degree of **Master of Engineering in Electronic Instrumentation and Control**, submitted to Electrical & Instrumentation Engineering Department of Thapar Institute of Engineering & Technology (Deemed to be University) is as authentic record of my own work carried under the supervision of **Mr. Pankaj Joshi** and **Dr Moon Inder Singh**. The matter contained in this dissertation has not been submitted, neither in part nor in full to any other degree to any other university or institute except as reported in text and references.



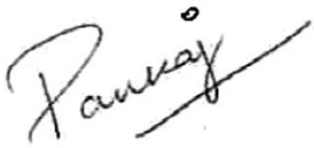
Place: Gurugram, Haryana — **Mandeep Singh**

Date: 14-Oct-19 — Roll No.: 801751002

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.



Dr Moon Inder Singh
Assistant Professor, EIED
Thapar Institute of Engineering and Technology
Patiala, Punjab



Pankaj Joshi
Senior Staff Engineer
Stryker Corporation
Gurugram, Haryana

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals in the organisation. I want to extend my sincere thanks to all of them.

I am highly indebted to Mr Pankaj Joshi, Mr Wiselka Matthias, Mr Anil Vishwakarma and for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I want to express my gratitude towards my parents & employees of Stryker Corporation for their kind co-operation and encouragement, which helped me in the completion of this project.

I want to express my special gratitude and thanks to industry persons for giving me such attention and their precious time.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

Abstract

In today's advanced Internet world having data and keeping it safe is what keeps one ahead of the competitors. This project is a sincere effort towards enabling all Stryker devices to gather data and route it to a central position where it can be stored and processed. Remote Diagnostic and Management Software is a tool that can be used by any Stryker device to connect with Stryker Cloud securely.

It provides a list of functionalities that allows the new software development teams to integrate it with their software efficiently. The RDMS resides inside the host device and runs independently of the master software. The host devices communicate with RDMS via different endpoints which are exposed locally. The motivation behind new development was that the earlier system had some drawbacks which were supposed to be removed and the addition of new features was to be done.

The primary purpose of the project is to add new features like being Operating System (OS)-agnostic, file downloading capability, cloud to device messaging, data exchange via Transmission Control Protocol (TCP) socket connection in the existing code. The project development was completed within six months using the Agile Software Development Life Cycle.

The development planned in two major phases, the first was to enable the RDMS to communicate with devices over TCP sockets and the second was to add the file downloading and cloud to device messaging features on the cloud connectivity side. After the completion of the development, the RDMS went through two phases of testing and released for internal use by the Stryker New Product Development (NPD) teams.

About Stryker Corporation

Stryker (NYSE: SYK) is included in Fortune five hundred lists, it is a firm with expertise medical technologies, and it operates from its headquarters at Kalamazoo, Michigan. Stryker's devices include implants that are used in surgical equipment and navigation of surgical equipment's, handling the patients and medical equipment used in emergency rooms, endoscopic and communications systems, joint replacement and trauma surgeries, spinal devices and neurovascular, neurosurgical, as well as other medical device products used in a variety of medical specialties. [1]

In the USA, most of Stryker's products are directly marketed to hospitals, doctors, surgeons and other healthcare facilities. Globally, Stryker devices are sold in about over 100 countries through company-owned sales subsidiaries and subdivisions and also via 3rd-party distributors and dealers.

The Orthopaedic Frame Company, the mother company of Stryker Corporation, was inaugurated in 1941 by Dr Homer Stryker, in Kalamazoo, Michigan. The company developed a Turning Frame, an adjustable hospital bed that gave access for easy mobility and repositioning of patients while also enabling necessary body mobility. The cast cutter, a new cast cutting tool that was used in getting rid of the cast material without damaging the soft skin tissues, and the walking heel, among many others. In the year 1964, the company name was officially renamed to Stryker Corporation.

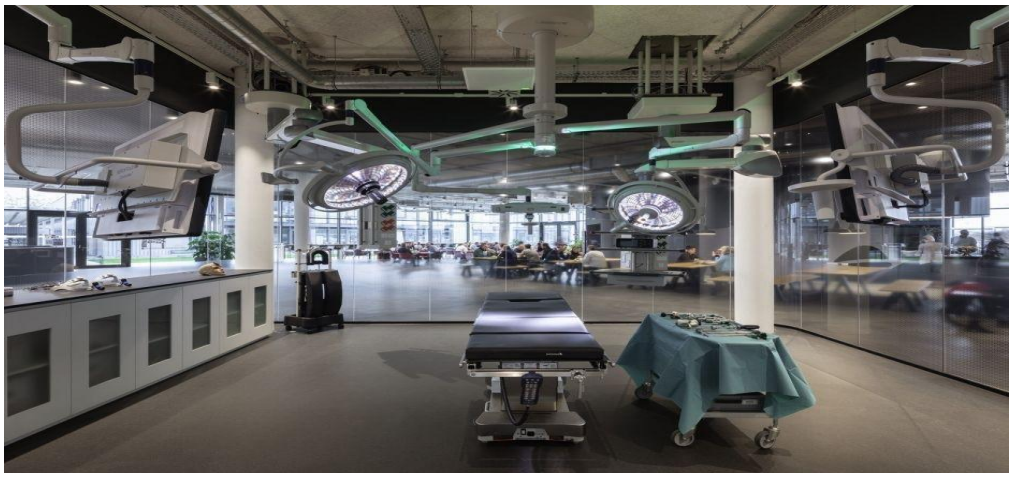


Figure 1 Inside View of Stryker

Chapter-1 Introduction

The Introduction of the project contains the Background of the project, old developments, the reason and motivation for the overall development of this project and the structure of the report.

1.1 Motivation

Stryker is a massive corporation with many business divisions, and every division has its range of products like endoscopes, artificial joints, advanced beds, power tools, smart emergency rooms and the long list will go on. As they are different business units, they have different cloud endpoints and different ways to connect to it and interpret its data. So, the primary motive of this project is to unite every product of Stryker Corporation to a single cloud endpoint.

By achieving this, one can have higher security as all the efforts will be concentrated on a specific piece of software and a single point where the data can be processed and interpreted. Having one secure cloud is the primary motivation behind the development of Stryker Remote Diagnostic and Management Software. So, the Division “Connected Care” took this initiative to connect Stryker Corporations’ Clouds. To do so, they required to have a piece of software which will securely transfer data to cloud storage, and not any cloud but a securely maintained cloud.

Stryker Remote Diagnostic and Management Tool was first developed in 2016. It had two cloud endpoints where one was a third party-maintained system and other was Stryker maintained cloud. RDMS had to be designed in such a way that it could be easily incorporated in their device’s software.

1.2 Background

Its development began in the year 2016 with the primary focus to allow a tool to quickly and securely upload their data to Stryker Cloud. The very first version was released after testing in the same year. Moreover, this software was ready to be used in both popular platforms, i.e. Linux and Windows. The first version was used in SDC3, i.e. a top-rated product of Stryker, which is used in Endoscopic surgeries. RDMS offered a list of functionalities like sending average data, and specific event occurrences, logging the device activity, uploading any files and many more.

The basic working of the RDMS is that it works in parallel to the devices original operating software just like an application running on windows, i.e. it is independent and uncoupled from the device hardware and software. Then the device's operating software communicates with the RDMS through various local endpoints that are exposed by the RDMS. These endpoints are secure and cannot be accessed by any external communication devices. The local parameters also do validation on the data that the device's software sends to it, which means it will not upload just any type of message the device has to send the messages to the local endpoint in a specific format only, else it return backs and error to device software to correct the form the message they sent. After the successful recognition of the message, the RDMS send the message to the cloud endpoint with the addition of header so that the identification of a message is accessible at the cloud side.

RDMS was maintained for over three years. During this period of maintenance, the development team saw some challenges and drawbacks which were noted and added to the features list of the second version of RDMS. These new features that are required to be added are listed in Chapter - 4 as Objectives of the project.

1.3 Organization of Report

The Structure of the report that how various chapters describe the steps involved in completion of this project

Chapter 2 - Contains the Literature Review that was required to be understood and appropriately learnt and a brief explanation of how the RDMS is structured and how it is operated.

Chapter 3 - Contains the Drawbacks of the first version of RDMS and their possible solutions that will be resolved in future or this version.

Chapter 4 - Defines all the main objectives that need to be achieved for the second version of RDMS.

Chapter 5 - Contains information about the work that has been done to achieve the goals and objectives.

Chapter 6 – Contains all the challenges that were faced during the project.

Chapter 7 – Has the outcome of the project and with a discussion about the period, it took to achieve the goals.

Chapter 8 – Finally, the use and the way various NPD teams integrate it discussed here.

Chapter 9 – Any further developments that can be done to make it more advanced.

Chapter-2 Theory

This Chapter contains the Literature Review material that is necessary to be understood and learnt before even knowing about the deeper aspects of the project, and this gives an overview and the working of the RDMS software.

2.1 Literature Review

The following were the basic works of literature that were required to be studied to understand and work on the project.

2.1.1 .NET Core

.NET Core is a multi-platform, open-license, free, and a managed piece of CLR framework for the Windows, Linux, and Mac OS. The project is majorly developed by Microsoft and can be used under the MIT License.

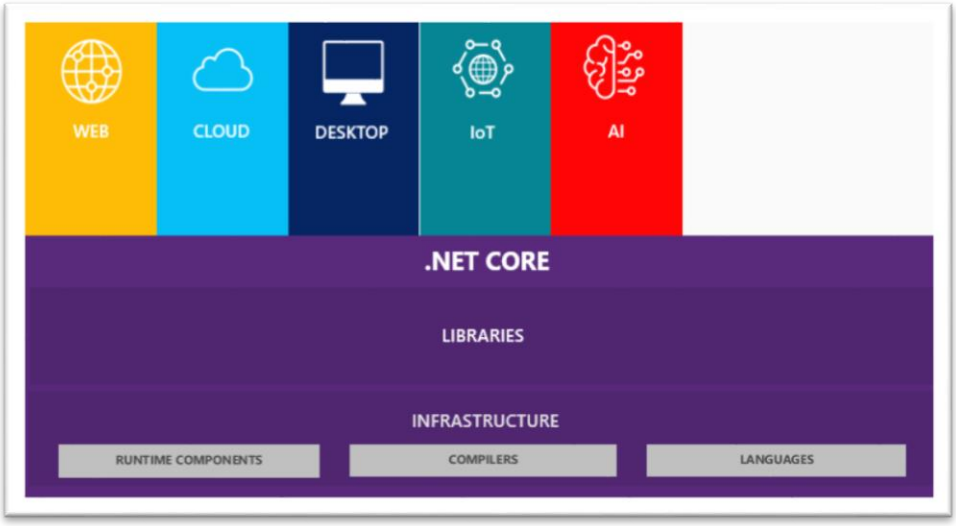


Figure 2 Structure of .Net Core

dotnet Core is a branch of the complete and feature-rich dotnet runtime and is cross-platform, and on top of that, it is free to use. In accumulation, dotnet Core is enhanced to execute on the cloud stage, such as Microsoft Azure, and is extremely modularised, with enhanced performance, and accepts a true self-sufficient positioning model. [2]

The dotnet Core repository can be changed to consider enhancing the performance of the software. When creating a branch, add the code optimisation and type it so that it can be known

to the public that it was performed. If the optimisation is a success, then it can be positioned into the source branch and be a part of the next release. If the optimisation is not successful, it would be possible to build and compile a release of the dotnet Core runtime specifically for that one software that requires.

By creating the dotnet Core runtime as open-source, Microsoft has genuinely involved the open community of designers and thinkers, giving them an enormous opportunity to rewrite, make a place for themselves, and use their existing skills to underwrite and develop their careers. [2]

dotnet Core is moveable, and it is entirely cross-platform yielding. It is also ascendable due to its minor footmark, evidently when linked to the complete dotnet Framework. Once a dotnet Core software is collected, only the binaries compulsory to run the software are wrapped into the gathering or executable, creation, reproduction and deployment are very easy. Because the code is a divergence of the complete dotnet Framework, which is a time-tested and deeply concocted programming library, the encryption is robust to temporary errors and handled exceptions.

dotnet Core provides many performance advances compared to the complete dotnet Framework. Several ideas for advancement have come from the open-source public. Now, as an alternative to working around performance issues, designers can interpret the source code producing the sluggishness and enhance it. Further, it is transported via a customary of NuGet packages. The “standard” assemblies are existing to take care of a scenario, for instance where a designer has a CPU that is not linked to the Internet and is consequently not able to transfer and install the elementary NuGet packages. Though, these defaulting assemblies safeguard the obstructions with making advancement once the dotnet Core 2.0 software is developed. [2]

2.1.2 C# Language

C# is a development language whose essential syntax looks identical and comparable to the syntax of Java. Though, calling C# a Java clone is mistaken. In real, both C# and Java are associates of the C household of programming languages (e.g., Objective C, C, C plus) and, thus, share a comparable syntax. The reality of the stuff is that numerous of C#’s syntactic concepts are displayed after various features of Visual Basic (VB) and C++. Like VB, C# wires the idea of class belongings (as different to old getter and setter approaches) and non-compulsory parameters. Like C++, C# lets one overwork operators, create assemblies, call back functions (via delegates), and enumerations.

C# supports several structures traditionally found in several functional languages (e.g., LISP or Haskell) such as lambda terminologies and unidentified types. Additionally, with the arrival of Language Integrated Query (LINQ), C# provisions a numeral of constructs that brand is unique in the software design landscape. Nonetheless, the wholesale of C# is indeed inclined by C-based languages. Since C# is a hybrid of abundant languages, the consequence is an invention that is as syntactically clean (if not disinfectant) like Java, is about as modest as VB, and delivers just approximately as much influence and elasticity as C++. [3]

It is significant to mention that the C# language could be used solitary to build software's that are compered under the dotnet Framework (could at no time use C# to shape a native COM server or a poorly managed C/C++-style submission). Formally speaking, the stretch used to describe the code aiming the dotnet Framework is managed code. The basic unit that comprises the controlled system is labelled an assembly (more minutiae on meetings in just a bit). Equally, code that cannot be unswervingly hosted by the dotnet Framework is termed unmanaged code.

As stated previously, the dotnet Framework can run on a diversity of operating systems. Thus, it is reasonably possible to develop a C# application on a Windows machine using Visual Studio as an IDE and deploy the program on a macOS appliance using the dotnet Core runtime. Also, one can develop a C# application on Linux utilising Xamarin Studio besides run the program on Linux, macOS and windows. With the latest release of Visual Studio 2019, can also build dotnet Core submissions on a Mac to be executed on Windows Linux, or macOS. To be precise, the concept of a managed situation makes it probable to develop, deploy, and implement dotnet programs on a wide diversity of target machinery.

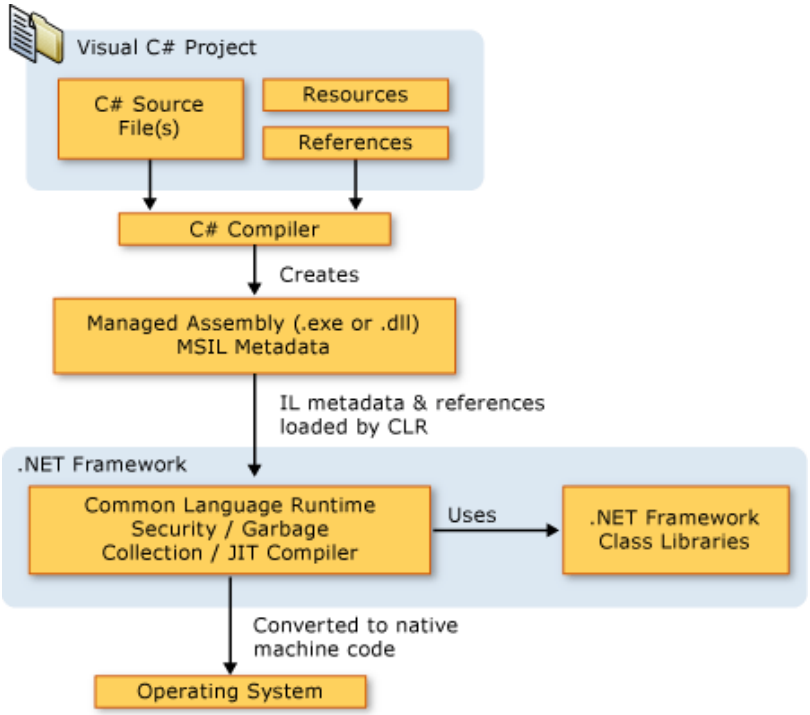


Figure 3 C# Language Structure

2.1.3 OOPS

Over the previous 18 years, the software design world engrossed on the expansion model of object-oriented programming (OOP). Most up-to-date advance environments and languages use OOP. However, OOP methods the foundation of everything of the progress today. This advancement will go into feature around the distinct characteristics of OOP and in what way they touch of the growth. [4]

OOP has transformed the way expansion is done. The know-how about objects, classes and the relationship between objects and classes are explained underneath. Also described the stages we need to gross when the arrangement of the classes and a few visual tools that can be used to achieve these steps.

Some of the key features that make OOP so famous are:

Eliminate Redundant Code

With the use of objects, one can decrease the quantity of code have to rewrite. If the system is engraved to design a receipt when a purchaser checks out, one will need that similar code accessible when there is a need to reissue a receipt. If the positioning of the system is to copy the revenue in the Sale class, one should resolve not to have to redraft this code another time.

This process does not only protect time but repeatedly helps remove blunders. If OOP is not used and there is a revolution to the statement (even to some degree as humble as a graphic modification), one can partake to make sure alter the PC and transportable applications. If there is slippage then any of them, one can track the danger of having the two interfaces behave differently.

Ease of Debugging

By understanding all of the code connecting to a method in one class, one recognises where to link when in runtime there is a problem with the code. It may not be a massive part of the application, but when the code reaches hundreds of thousands or even lakhs of lines of code, it resolves to save an ample of time.

Ease of Change

To rearrange all of the code in a class, then as an effective change in the submission, one can alter the classes and deliver a new class with totally dissimilar functionality. Though, the modified class can still intermingle with the rest of the software in an identical way as per the existing class. It is like car parts. If the desire is to swap a muffler on a car, this does not require to get a new car. If the code linked to the task is scattered all over the place, its kinds it much more tough to transform items around a class.

Figure 4 Features of OOP

OOP has these two major components:

1. **The Object -** An object is no matter what that can be represented. To better comprehend what a software design object is, the first aspect is a few items in the real world around us. “A physical object can be anything around you that you can touch or feel”. Let take

an example, a television. Televisions also have roles. They can be twisted on or off. One can alter the channel, regulate the volume, and modify the brightness. [4]

2. **A Class -** A class is essentially a cookie-cutter that could be castoff to make objects that have comparable features. All items of a specific class will take the same assets (sign, the principles of the properties numerous times will be dissimilar) and the similar methods. The laws of those assets will alter from object to object.
    i. Planning Classes - Arrangement of our classes is solitary of the most vital steps in the expansion course. Whereas it is conceivable to go posterior and add possessions and methods after the detail (will undoubtedly require doing this), it is significant that to recognise which classes are working to be second-hand in the submission and which elementary properties and approaches they will take.
    ii. Planning Properties - Introduction this info in a class moderately than hardcoding it in the software will let to make deviations to this information in the upcoming years.
    iii. Planning Methods - The objective is not to add all of the approaches now, but the extra planning can be done at the launch, the calmer it will be easier to manage. Not each of the classes will have countless methods. Some may not have in the leastways at all. When scheduling the processes, recollection having the emphasis on a precise task. The more detailed the methods, the more likely it is that it could be reused.

2.1.4 Software Development Life Cycle

Software Development Life-Cycle (SDLC) is a basis that gives the steps included in the advancement of a piece of software at every phase. It contains the overall strategy for development, deploying, testing and maintaining the software. [5]

SDLC gives steps for the whole cycle of progress. Following are the various phases of SDLC:

1) **Requirement Analysis and Gathering**
    In this stage, all the significant and related info is gathered from the client to grow an artefact as per their prerequisite and expectancy. Any distrust or doubts must be determined in this stage only. Before starting a product, a good understanding, and familiarity with the product is very vital. SRS document is issued. This document

should be carefully understood by the designers and should be studied by the client for future reference.

2) **Design**

During this stage, the incline of structures to be involved are collected in the SRS document, and it is used as an effort and software architectural strategy document that is used for employing system progress is built.

3) **Coding**

The advance of the code begins once the designer is assigned the Design document. The Software plan is rendered into source code. All the mechanisms of the software are employed in this stage.

4) **Testing**

Testing begins once the development is completed and the code is free for trial. In this stage, the established piece software is tested carefully, and any flaws found are handed over to developers to get them fixed.

5) **Deployment**

Once the technologically advanced product is carefully tested, it is arranged in the required manufacture environment, or primary UAT (User Acceptance testing) is executed dependent on the client prerequisite and expectancy.

6) **Maintenance**

After the deployment of the software on the manufacturing environment, conservation of the software is taken attention of, i.e. if any severe issue arises up to and that requires to be resolved, or any augmentation is to be achieved taken care by the software developers.

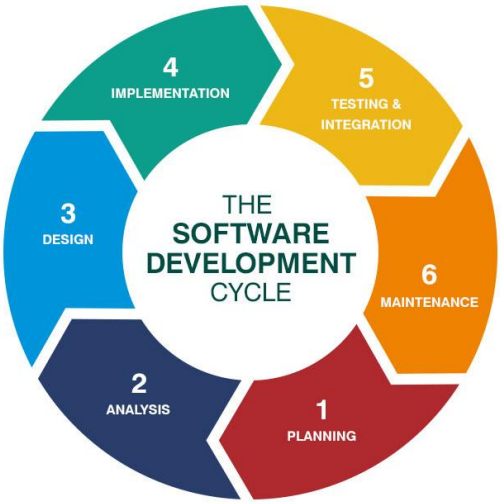


Figure 5 Software Development Life Cycle

In the Development of this project, the Agile SDLC was adopted. It is explained below:

Dexterous method of intelligent had originated precisely on time in the software enhancement and started getting to be markedly in fashion with stretch because of its adaptability and elasticity. An iterative method is reserved, and working software design theory is agreed on after every cycle. Each procedure is incremental as far as characteristics; the last form has every one of the highlights committed by the client. The most excellent valued deft practices incorporate "Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995)". [5]

Following are the stages standards:

1. **Entities and associates** - In the deft procedure, self-association and enthusiasm are overbearing, as talk like co-area and competition software design.
2. **Effective programming** - Showpiece working program design is seen as the greatest asset of communication with the customer to understand their state, rather than merely to rely on papers.
3. **Client matched effort** - As the requirements cannot be collected totally at the term of formation of the responsibility because of unlike perspectives, reoccurring client announcement is significant to get suitable item requisites.
4. **Answering to change** - Dexterous development is concentrating on a fast comeback to alteration and persistent growth.

5. Functionality can be shaped speedily and crisscross one succeeding to the supplementary.
6. It is a good model for the situations which can transform rationally.
7. Slight or no positioning required, upgrading and association are accomplished next to each other.

Moreover, there are some drawbacks of the Agile methodology

1. Tough to retain up and accomplish the requirements, in bright of the fact that any of the development is not completed actually.
2. Credentials are less subsequently singular circumstances are prolonged
3. Be contingent powerfully on teamwork with the client, so if the client is not clear, then it will style equivocalness amongst the concocts.

Comparison between two famous SDLC models waterfall and Agile refer to table 5

Table 2 Comparison between Waterfall and Agile

Model/ features	**Waterfall model**	**Agile model**
Requirement specifications understanding requirements	Beginning well understood	Frequently altered well understood
Cost Guarantee of success	Low Low	Frequently altered Well understood
Resource control Cost control	Yes Yes	Very High Very High
Simplicity Risk involvement	Simple High	No Yes
Expertise required Changes incorporated	High Difficult	Intricate Reduced
Risk analysis User interaction	Only at the beginning Only at the beginning	Very high Difficult

Overlying phases	No phase	Yes
Maintenance	Least glamorous	Promote maintenance
Flexibility	Rigid	Highly flexible
Integrity & security	Vital	Ability Obvious
Reusability	Limited	Reusable
Interface	Minimal	Model determined
Training required	Vital	Yes
Time frame	Extended	Minimum possible

2.1.5 Transmission Control Protocol

In the Internet Protocol (IP), computers send messages to other computers via a network of routers. However, there is a limitation to how big those messages can be, due to the physical limitations of the internet's infrastructure. That is why computers split those messages into multiple small packets. [6]

Many things can go wrong when a single message is split into multiple packets. Here are some big ones:

- Packets can be lost, due to problems in the physical layer or routers' forwarding tables. A message needs every single packet to be put back together.
- Packets can arrive out of order. That can happen, especially if two packets follow different paths to the destination.
- A computer might send multiple messages to a destination, and the destination needs to identify which packet belongs to which message.

So, TCP is the solution, and the Transmission Control Protocol is always used with the IP to make sure of reliable transmission of packets. Together, the protocols are named as TCP/IP. Now we step through the whole process of receiving and sending packets. The notable point is that the IP routing protocol governs this process.

Step 1: Establish a connection

When two computers want to send data to each other, they first need to establish a connection with the TCP "3-way handshake".

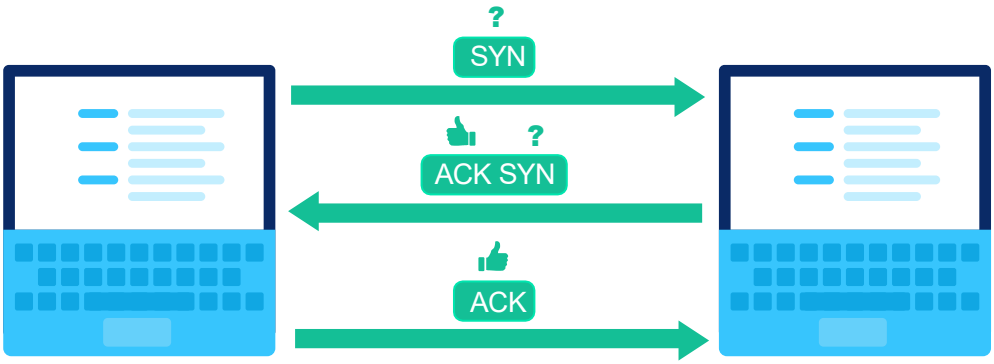


Figure 6 TCP connection establishment

Diagram of two computers with arrows in between.

- The message goes from device 1 to device 2 with the "SYN" label.
- The message goes from device 2 to device 1 with "ACK SYN" label.
- The message goes from device 1 to device 2 with the "ACK" label.

The first device sends out an SYN message (SYN = "synchronise?"). The second device sends back an ACK message (ACK = "acknowledge!") and another SYN. The first device replies with another ACK. Once the devices are done with this handshake, they know they all are ready to receive and send actual data.

Step 2: Send packets of data

Either computer can now send packets of data. When either computer receives a packet of data, it always replies to confirm what it received.

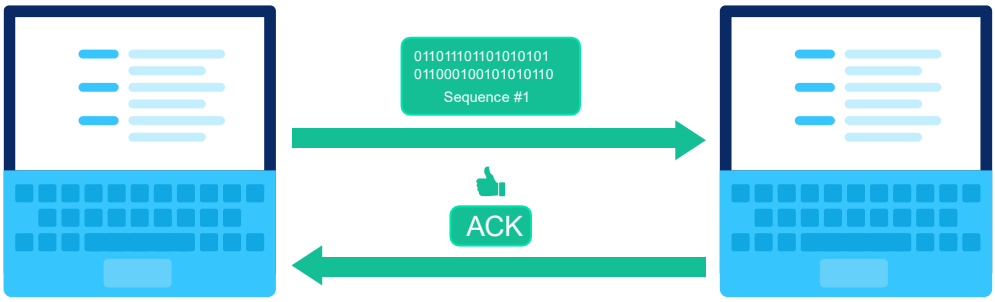


Figure 7 Sending packets over TCP

Diagram of two devices with arrows in between.

- The message goes from Device 1 to Device 2 and shows a box of binary data and the label "Sequence #1".
- The message goes from Device 2 to Device 1 with "ACK" label.

The first device sends a message of data and labels it as the first in the sequence. The second device acknowledges receiving it with an ACK. Each message contains the data itself, plus necessary headers that will improve the reliability of TCP, e.g. the sequence number.

Step 3: Close the connection

Either device can close the connection when they do not want to send or receive data.

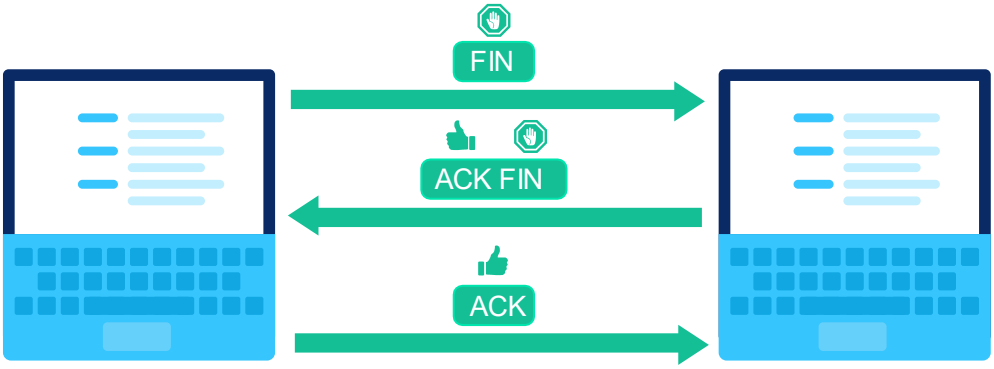


Figure 8 Closing TCP Connection

Diagram of two Devices with arrows in between.

- The message goes from Device 1 to Device 2 with "FIN" label.
- The message goes from Device 2 to Device 1 with "ACK FIN" label.
- The message goes from Device 1 to Device 2 with "ACK" label.

The first device initiates closing of the TCP connection by sending a FIN message (FIN = finish). The second device replies with an ACK and another FIN. Finally, one more ACK and the connection will be closed.

Reliable transmission

TCP includes multiple types of mechanisms that will ensure reliable transmission of packets.

As one example, a sensing device can detect when a packet has disappeared by noticing that a packet has not been acknowledged within a specified period. Once the sender detects this, it will re-send the lost packet.

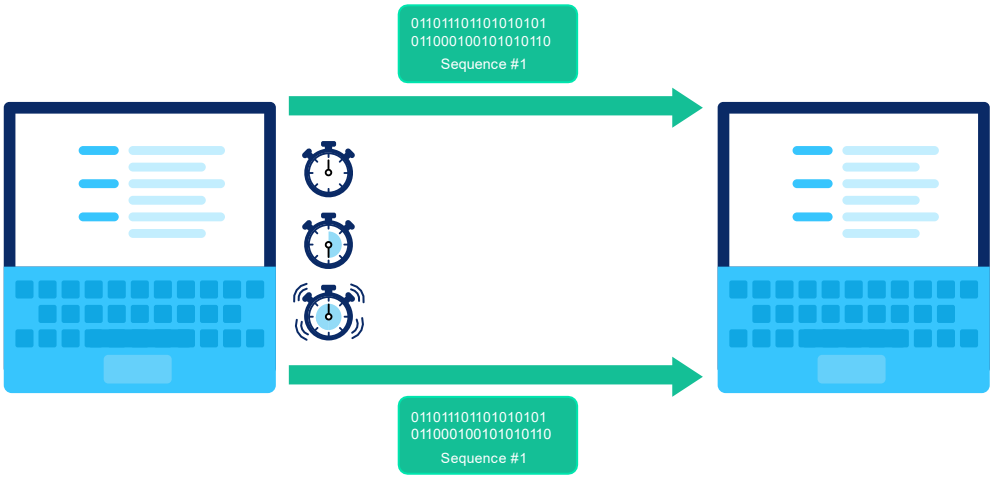


Figure 9 TCP Data Resending after a timeout

2.1.6 IoT (Internet of Things) Hub

Be definition “The Internet of Things refers to devices that collect and transmit data over the Internet”. These devices can be whatever from TV or Fridge to Bed or wearable devices such as the Apple Watch, Microsoft Band, or Fitbit, to name a few. It is said that if a device has a power switch, produces data, and can connect to the Internet, the odds are that it can be a fragment of the IoT. Today, most cars are sending data to the Internet. By the year 2020, it is assessed that over 300,000 cars will be linked to the Internet. The wearable device market cultivated 223% in 2015 worldwide. Many corporations are participating in home control devices, such as Samsung. It is projected that by “connecting” kitchens to the Internet, the beverage and food business could save as plentiful as 20% annually. According to a few evaluations, the Internet of Things will add $15-$20 trillion to the worldwide GDP in the upcoming ten years. [6]

Supplementary prevalent devices that are rummage-sale today to produce data are the low price, small-sized supercomputers such as the Tassel IO, Raspberry Pi, and Pi Zero boards. These tiny devices are regularly used to generate and make data in situations and places where standard laptops and PCs are not capable enough or cannot drive. In a circumstance, it is these mini-computers that will be used in future.

Though the term Internet of Things was created in 1999, the notion of gathering and communicating data extends back to earlier than 1999. ATMs started using it since 1974. Still, with the current program and enthusiasm around data and IoT, IoT is till now a new perception to many folks.

IoT Hub also supports bi-communications both from the cloud to the device and from the device to the cloud. IoT Hub also gives support multiple messaging forms such as file upload from devices, device-to-cloud telemetry, and request-reply methods to manage devices from the remote location. IoT Hub Realtime monitoring helps one maintain the health of device by tracking all the minute events such as device failures, device creation and device connections. [7]

Numerous key aspects are serving to drive the IoT era, and one is the low-cost costs of the machinery. For example, both the Tassel IO and Raspberry Pi board are fewer than $40, and the accompanying instruments and components, such as climate, GPS, ambient modules, and accelerometer are even cheaper. Separately of these small-sized boards are powerful enough to run cutting-edge software, such as Windows 10 Core. There are new boards as well, with ones from Arduino and Intel.

IoT Hub is a well-organised service, basically, a computer running in the cloud, which turns as a central connection hub for bi-directional communication medium between IoT application and the computer it manages. IoT Hub can be used to develop IoT solutions with secure and reliable communications between thousands of IoT devices and a remotely hosted solution backend. Also, connections can be made virtually with the device to IoT Hub.

Another aspect is the improvement in software that delivers amusing, active, and high-end data analysis and investigation skills. The software is fetching more authoritative features and calm to use, giving the capability to bring the enterprise-class and high-end analytical vast data resolution needs. [6]

The explosion of wireless and cellular connections has also increased IoT solutions, letting important information and IoT resolutions to contain mobile apparatuses and provider networks to the Internet where beforehand terrible. As making a connection to the Internet remains to advance and turn out to be less of a cost block, the surge in IoT circumstances has enhanced enormously.

Finally, a crucial central aspect in the bang of IoT situations has been the quick development in cloud amenities and knowledge, providing an extremely cost-effective resolution for data management, analysis, and storage. Attached with the statistic that the current expansion tools and machinery used on-premises also labour when evolving for the cloud brands using cloud-based amenities and resolutions highly beneficial.

IoT Hub also supports bi-communications both from the cloud to the device and from the device to the cloud. IoT Hub also gives support multiple messaging designs such as file upload from devices, device-to-cloud telemetry, and request-reply methods to control the devices from the remote location. IoT Hub Realtime monitoring helps to maintain the health of the device by tracking all the minute events such as device failures, device creation and device connections.

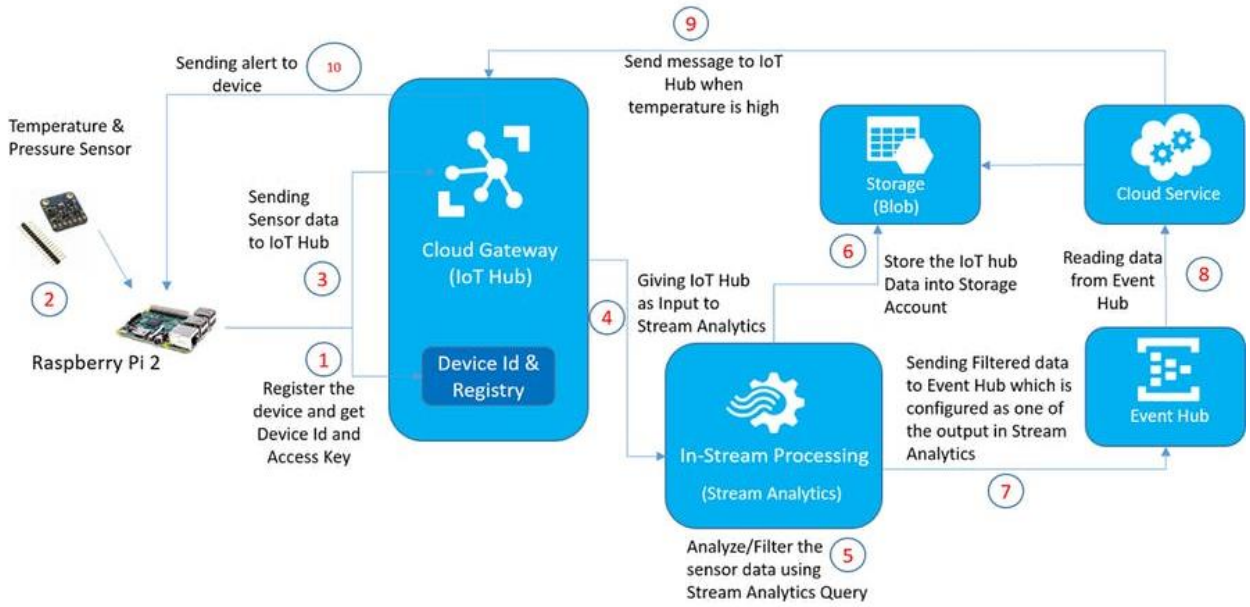


Figure 10 Example of device integrated with IoT Hub

IoT Hub's capabilities help to develop scalable, full-featured IoT solutions, for example, managing industrial devices used in manufacturing, tracking valuable assets in any field, and observing office building usage.

2.2 About Software RDMS

It is an overview of the Remote Diagnostic and Management Software

2.2.1 Overall Working

RDMS was developed using dotnet Framework in the year 2016. It had some basic features that are listed in 2.2.2. For RDMS to work, it has to be installed inside the Hosting device in parallel to the leading software that controls the device. For example, assume there is a device which has windows as its Operating System and that every device has its window services to keep that device running.

Now the RDMS is to be installed as a service parallel to those primary services running in the device. Also, now the core services will communicate with the RDMS through a local communication protocol. After receiving a successful request from those local protocols, it validates the request and sends it to the cloud.

The device on which the RDMS runs is called the host device. The main advantage here is that all the software development teams do not have to implement the same thing over and over, they need to install the RDMS and deliver their proprietary messages to RDMS and rest will be taken care by it.

Figure 11 RDMS setup in Host

2.2.2 Features

RDMS provides a list of features as stated below

1. It allows a device to send any **Event** occurrences to the cloud, e.g. Light turned ON.
2. It allows devices to send any format of **Data** to the cloud, e.g. Temperature value.
3. It allows to **Access** both 3rd party clouds as well as Stryker Cloud.
4. Devices can upload any size and type of **File,** e.g. sending picture inside of the stomach.
5. Allows to **Log** devices activity and upload at the desired time.
6. Allows the device to **Process** the data before sending.

2.2.3 Structure and Functionality

Now we know that the RDMS resides inside the Host Device and Host devices services communicate with the RDMS, also what functionalities it offers. Now we try to understand how it is achieved. The host services or we can use another term called client will send a message to the RDMS. These messages have a defined format, and if a client tries to send a message with invalid values and parameters, the RDMS will discard that message.

Also, not everyone can send that message to the RDMS; only the client that is in the local machine can communicate to the RDMS. It exposes a local endpoint to the client on which it

can put its message on, and this message will be picked up the RDMS also it will send back a confirmation to the client that its message has been successfully validated and will be sent to the cloud.

After the RDMS has successfully validated a message, it will go through a series of modifications and will be sent to appropriate cloud endpoint through a secure connection. It maintains an internal queue in which all the messages will be queued and sent one by one.

Chapter-3 Drawbacks of Initial Versions

After the use of version one, there were feedbacks from the NPD teams which are discussed below

3.1 Operating System Dependency

The first Drawback that was observed was its framework dependency. The dotnet framework is designed to run on Windows machines only. So, in order for the devices with Linux as an Operating System, the NPD teams had to install ‘mono’ a third-party framework which will enable the dotnet framework apps to run in Linux. Due to installing ‘mono’ the size of the package increases, which is not desired by the development teams.

3.2 DoD Restrictions

For the medical devices which are used in Military Hospitals, there are strict guidelines by the Department of Defence that there should not be any open Web Hosts in the device. Because from a security point of view a hacker can access that local web host.

As the RDMS exposes a local web host for the client to get the messages, this was a significant drawback. So, an alternative way to communicate with the RDMS is required if the devices are to be used in military hospitals.

3.3 Missing File Downloading

A common thing these days but was a significant drawback in RDMS core code. With the file download functionality, the NPD teams could push updates or security patches or get some files that they want from the server. RDMS had the functionality to download files from the third-party cloud but not from the Stryker Cloud. As in the coming future, the 3rd party cloud was to be ruled out, so this functionality was lacking the RDMS.

3.4 Back Channel Missing

The RDMS had only the functionality to send the message from the device to the cloud, but it clouds not receive ant messages from the cloud, which was a significant drawback as the NPD teams wanted to use that to push notifications to Device and to control the device remotely.

3.5 Security

Security is a significant feature that everyone looks upon when transferring data to a remote location. As RDMS was developed in 2016 and there were updates in the security parameters, this was a significant demand of the NPD teams. The Security that RDMS was using was not up to date as being 3-year-old there were newer ways to connect to the cloud securely. So yes, it was a drawback in the initial version.

Chapter-4 Objectives

The Objective that was required to be completed in this project are as follows

1. Make the Tool Operating System agnostic
2. Add file/update downloading functionality.
3. Add TCP as a medium for connection with Tool.
4. Enable backchannel forwarding messages from cloud to device.

Chapter-5 Work Done

In this chapter, we will discuss how the project objectives were accomplished. Moreover, all the development done is used and released as version 2 of RDMS.

5.1 Migration to .NET Core

So as per the first objective, the RDMS should be framework independent the version one was made on dotnet framework, and now we had to move the code to dotnet Core, as studied earlier it is framework independent. However, the major challenge is the libraries used. Since earlier it was in Dotnet framework and it has different libraries and different setup.

So, new libraries were searched for best fit and all the code was migrated to Dotnet core with advancements. Overall the code structure and the logic implemented remained the same, only thing that was required to be taken care of was that new libraries that we use should behave that same as the previous ones used or if the libraries work different then at least the overall functionality of the code should remain the same, i.e. refactor the code if required.

5.2 Addition of Socket Interface

After successfully migrating the old development, the second objective was to make it DOD compatible. As per the Department of Defence security requirements, there should not be any open Web server hosted locally on any device. So, in order to get an alternative way to connect with the RDMS a socket connection was introduced which would use Transmission Control Protocol to transmit all the messages.

Now the messages could be just put on a TCP stream after establishing the connection with RDMS. Using Socket connection to transfer messages also gave an additional functionality which was unavailable earlier in the web hosts because now the RDMS could initiate the messages and the client can receive them synchronously. So, the socket functionality was added with all the validations and secure transfer of messages.

5.3 File/Update Download Functionality

The first version did not have this functionality through the Stryker cloud, and to achieve this IoT Hub was used in which the file was first uploaded on the cloud to secure space. Also, whenever the Device makes a connection with the internet, the cloud operator pushes the specific download to the RDMS.

Now the download process starts where the RDMS regularly updates the cloud operator about the progress of the download. So it also has a pause and resumes feature like when the Device wants to stop massive network operations, it can pause the downloads.

After the successful completion of the download, the RDMS will notify the device and then the device and take care of the download from there, whether it needs to be just stored or it needs to update the packages inside.

5.4 Enabling Cloud to Device Messages

The major drawback in the version one was that it could not send cloud to devise messages because of two reason fists, as there was not cloud operating tool which will enable to generate a message from the cloud and the second, was the web hosts, they have a limitation that the server can not initiate anything the client has to request, and the server will respond.

As now with the new TCP interface, there was a back channel available, and now the RDMS could easily transmit the messages coming from the cloud down the device client which is listening to them. With the addition of this feature, the remote operator could easily change the configuration or give a command or even take control of the device.

Chapter-6 Challenges

6.1 Learning the language

The first challenge faced was adapting to an IT environment. The very first task that was assigned was to learn a new language C#. Being from an Instrumentation background and learning all the concepts of programming from scratch was a massive task in itself and becoming more proficient in it was another. It took like 3-4 months to become comfortable with the language. This learning process involved doing several minor projects.

6.2 Validating Messages

The first task in the project was to add the socket interface. The socket interface had to be up all the time and should parse all the values given to it and return any errors if the costs are incorrect. The major challenge was in implementing this, and different threads were used to make it fast, e.g. one for receiving, one for sending and one for processing messages.

6.3 CPU Over Usage

The issue was observed during the development the CPU usage would go to 100%, which would cause the whole system to go slow as RDMS was consuming all the bandwidth. It was hard to debug issue but was resolved by as it was caused due to frequent reconnects on the socket side, So, basically, the RDMS was not disposing of the Threads that were generated during the connection.

6.4 Enabling Downloads

After getting over with the socket implementation, the next big task was enabling the downloads from the cloud. The logic to download seems easy, but it is very complicated when we have to implement it from scratch. There were so many challenges where the network could switch, and the speed cloud slows down, the file could be removed from cloud, the security was the main issue because the downloads are required to be safe, so a check is maintained after the download completed.

6.5 Resuming Downloads

It was another huge challenge to give functionality to the device to actively pause the download. It was a challenge because we had to pause without losing the progress and then resume it from the same position when prompted.

6.6 Connection Stability

The RDMS had a persistent issue of losing the connection with the secure cloud if the network got disconnected for a while. So, this was resolved by adding a heartbeat mechanism where the RDMS would continuously ping the cloud and keep track of the connection. If the connection were lost, it would establish a relationship with the secure Stryker cloud.

6.7 File Detection

The functionality of RDMS to give upload file when placed in a particular folder was not working correctly. It was a significant challenge faced in the bug fixing phase; it was because of the slow file writing and reading speed of the file system. The mechanism used to detect the file was not correctly working as it was detecting multiple files for one file. It was resolved using a wait that was configured according to the size of the file, i.e. greater the file size more considerable the delay in initiating the trigger.

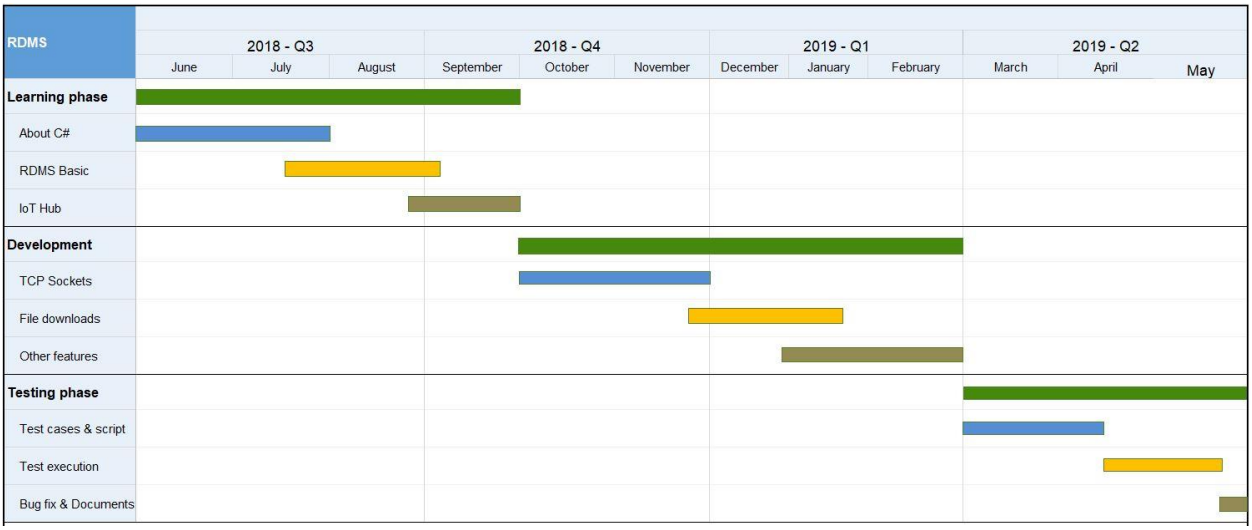
Chapter-7 Result and Discussion

The final results are as follows

- The basic requirements to understand and develop the software were met in the first four months.
- The development of the software with resolving all the challenges was completed in 5 months
- The code was reviewed continuously during development by the Cloud Tech-Lead.
- The Testing was completed within two months by experienced testers.

The project with the timeline is as follow:

Table 3 Gantt Chart for RDMS release



The development was divided into three major parts

1. Learning Phase – It was completed in 4 months
    i. The first part was doing mini projects to learn about the language used in RDMS
    ii. Then the basic working and structure of RDMS were understood.
    iii. Finally, the new functionality that was to be added was learned and practised.
2. Development Phase – It was completed in 5 months
    i. In development to get a better understanding and natural development first step chosen was to implement the TCP interface that would talk with the Core of RDMS

ii. The primary functionality of the RDMS, i.e. file downloading was implemented in the second phase of development where another feature was pausing the download on devices requests.

iii. Other features were added like configurable turning on/off of the cloud endpoints and many more minor features.

3. Testing Phase – It was completed in 3 months span

i. The first step was for the testers to understand and write the test scenarios for the RDMS. It was a critical part as the quality of the RDMS would have been dependent on the variety of the test cases.

ii. The second part of the phase was actual testing.

iii. Final was the bug-fixing stage where all the bugs were fixed, and the product was released.

Chapter-8 Conclusion

During development, the NPD teams were provided with the beta build versions to allow them to integrate RDMS in their devices so that if any requirement that might be required by the NPD could be incorporated before the final release of RDMS.

The overall conclusion of the project was that the RDMS was almost fully integrated and tested by the NPD teams with a lot of suggested improvements by them. Their project was based on the Linux environment, and the RDMS was running independently, uncoupled from the original software and with successful transmission of the messages.

Also, another team did a POC with pushing security patches and updating their firmware using RDMS download functionality. So also adding person profiles on their device remotely in runtime using the backchannel from the cloud, i.e. the cloud to device message functionality of RDMS.

Chapter-9 Future Scope

9.1 PKI Security

Public Key Infrastructure (PKI) is a safety apparatus for assuring that on-line infrastructures are authentic and cloistered. It is acquisition recognition as a means of realising secure e-commerce, thereby fighting one of the major worries about doing occupational online. [7]

Encryption is an influential tool for shielding confidentiality, but without that, data can be decrypted, it is persistently useless. Defining who has the authority to decrypt data and to access applications becomes the grave issue. This feature can be another advancement that this software can include.

References

[1] J. Brown, “Growth And Innovation In Medical Devices: A Conversation With Stryker Chairman John Brown,” *Health Affairs,* vol. 1, p. 3, 2007.

[2] B. Perkins, J. V. Hammer and J. . D. Reid, Beginning C# 7 Programming with Visual Studio® 2017, 10475 Crosspoint Boulevard, Indianapolis, IN 46256: John Wiley & Sons, Inc, 2018.

[3] A. Troelsen and P. Japikse, Pro C# 7: With .NET and .NET Core, USA: Apress, 2017.

[4] S. Kaczmarek, B. Lees and G. Bennett, Swift 5 for Absolute Beginners, Apress, 2019.

[5] S. Nazia, “Awareness Regarding Importance of SDLC among IT Professionals and Students,” *International Journal of Computer Applications,* vol. 1, no. June p. 5, 2019.

[6] D. N. m. D. s. and. B. A. K. , “A Study on SDLC For Water Fall and Agile,” *International Journal of Engineering & Technology,* p. 13, 2018.

[7] R. S. Larson, Bioinformatics and Drug Discovery, Ohio: Humana Press, 2005.

[8] S. Klein, IoT Solutions in Microsoft’s Azure IoT Suite: Data Acquisition and Analysis in the Real World, Redmond, Washington, USA: Apress, 2017.

[9] J. Barnes and B. Familiar, Business in Real-Time Using Azure IoT and Cortana Intelligence Suite, Chicago: Apress, 2017.

[10] J. G. Dumas, P. Lafourcade, F. Melemedjian and P. Thoniel, Qualitative Research in European Migration Studies, USA: Springer, 2019.

Thesis ME

ORIGINALITY REPORT

10%	4%	9%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

4%

★ Scott Klein. "IoT Solutions in Microsoft's Azure IoT Suite", Springer Nature, 2017

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		