Analysis of Agile Practices within Siemens Regarding Digitalisation of Internal Services for Client-Based Software Development Projects

A Dissertation Presented

By

Submitted to The University of The West of England in partial fulfilment of the requirements for the degree of:

Software Engineering for Business

April 2020

Department of Environment and Technology – Computer Science and Creative Technologies

# Abstract

*Working in an Agile environment has become commonplace among many organisations; however, the specific framework employed is unique to the needs of the company. Siemens is an established multinational organisation with nearly 400,000 employees. As such adopting Agile into the working culture proved beneficial and has drastically improved various aspects of this business, particularly in software development. The following report outlines the methodology employed using the Agile practices to achieve the development of a financial control application and details the advantages and pitfalls accompanying this process.*

# Table of Contents

[Abstract 2](#_Toc38542337)

[Table of Contents 3](#_Toc38542338)

[Abbreviations 5](#_Toc38542339)

[1.0 Introduction 6](#_Toc38542340)

[1.1 Project Summary 6](#_Toc38542341)

[Executive Summary 6](#_Toc38542342)

[Project Overview 6](#_Toc38542343)

[Overview of My Role 8](#_Toc38542344)

[2.0 Application Design 9](#_Toc38542345)

[2.1 Defining requirements 9](#_Toc38542346)

[Proposal from project team 10](#_Toc38542347)

[SOMO proposal to SFS 11](#_Toc38542348)

[Realisation of User stories 11](#_Toc38542349)

[2.2 Evolution of Requirements 11](#_Toc38542350)

[Proposal from project team: 11](#_Toc38542351)

[SOMO proposal to SFS: 12](#_Toc38542352)

[Realisation of User Stories: 14](#_Toc38542353)

[2.3 Functional Requirements 14](#_Toc38542354)

[2.4 Non-Functional Requirements 17](#_Toc38542355)

[Functionality 17](#_Toc38542356)

[Reliability 18](#_Toc38542357)

[Usability 19](#_Toc38542358)

[Efficiency 19](#_Toc38542359)

[Maintainability 20](#_Toc38542360)

[Portability 20](#_Toc38542361)

[2.5 Overview of Requirements gathering within Siemens 21](#_Toc38542362)

[Functional Requirements 22](#_Toc38542363)

[Non-Functional Requirements 22](#_Toc38542364)

[2.6 Requirements Traceability 22](#_Toc38542365)

[3.0 Project Management Within Siemens 24](#_Toc38542366)

[3.1 Overall Project Management Within Siemens IT 24](#_Toc38542367)

[Kanban 24](#_Toc38542368)

[Regular meetings 24](#_Toc38542369)

[Providing customer satisfaction through continuous delivery 25](#_Toc38542370)

[Critical Reflection 25](#_Toc38542371)

[3.2 Project Management for Software Development 25](#_Toc38542372)

[Combining frameworks 26](#_Toc38542373)

[3.3 Project Management for the Finance Application 26](#_Toc38542374)

[Commencing a Project and Defining User Stories 26](#_Toc38542375)

[Maintaining User Stories and Using Kanban 28](#_Toc38542376)

[3.3 My Role 29](#_Toc38542377)

[4.0 Agile Working Methods 30](#_Toc38542378)

[4.1 Level of Documentation 30](#_Toc38542379)

[Siemens Approach to Documentation 30](#_Toc38542380)

[Documentation for FFM and its relevance 30](#_Toc38542381)

[Critical Reflection 31](#_Toc38542382)

[4.2 Sprint Cycling/Working Practice 32](#_Toc38542383)

[Overview 32](#_Toc38542384)

[Sprint cycling within Siemens 32](#_Toc38542385)

[Critical Reflection 32](#_Toc38542386)

[5.0 Testing 34](#_Toc38542387)

[5.1 Application testing within Siemens 34](#_Toc38542388)

[User Acceptance Testing 34](#_Toc38542389)

[Pilot Testing 36](#_Toc38542390)

[Testing during development 37](#_Toc38542391)

[Unit Testing 38](#_Toc38542392)

[Critical Reflection 38](#_Toc38542393)

[5.2 My Role 38](#_Toc38542394)

[6.0 Conclusion 40](#_Toc38542395)

[7.0 References 41](#_Toc38542396)

[8.0 Appendix 44](#_Toc38542397)

# Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Explanation** |
| **SFS** | *Siemens Financial Services* |
| **SOP IT** | *Siemens IT* |
| **CFO** | *Chief Financial Officer* |
| **ARE** | *Account Region* |
| **BU** | *Business Unit* |
| **KC** | *Key Control* |
| **PCMB** | *Policy Control Master Book* |
| **FFM** | *Finance Future Makers* |
| **MLP** | *Minimum Lovable Product* |
| **RIC** | *Regional Internal Control* |
| **ISO/IEC** | *International Organisation for Standardisation/International Electrotechnical Commission* |
| **REST** | *Representational State Transfer* |
| **PKI** | *Public Key Infrastructure* |
| **UAT** | *User Acceptance Testing* |

# 1.0 Introduction

## 1.1 Project Summary

### Executive Summary

Siemens is a multinational technology and manufacturing company. The company is split into distinct divisions, each responsible for different operations. For example, Siemens Financial Services (SFS) and Siemens IT (SOP IT), which handle the organisation’s internal financial infrastructure and IT demands, respectively.

SFS partnered with SOP IT to undergo a digitalisation project. This project is part of a broader internal drive for digitalisation and aims to bring current manual processes into the digital workplace, this is being accomplished primarily through a Financial Services application.

My aim for being part of the project team was to gain an insight into how Siemens apply Agile within their working practices; I was able to monitor these methods in order to critically evaluate them.

### Project Overview

An initial project team was assigned the challenge of digitalizing SFS. A team within this division determined that the processes that they and their peers were undertaking were laborious and antiquated. They concluded the most appropriate means to improve this was to develop an application to make work less manual and therefore, more efficient. Based on the experiences of the SFS team members, a set of base requirements was drafted; detailing what improvements the financial processes the application must adhere to (*see Figure 1.1*).

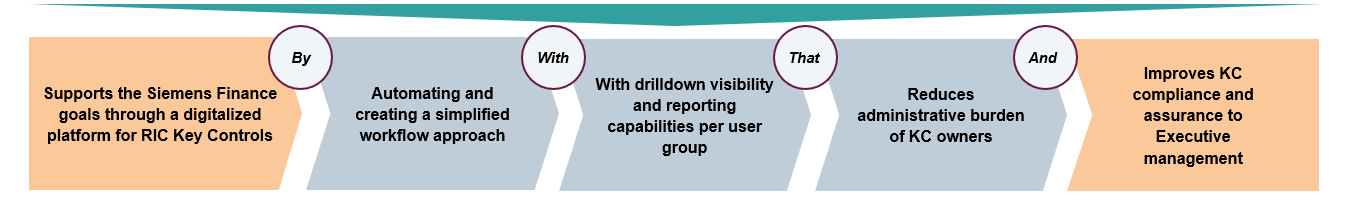
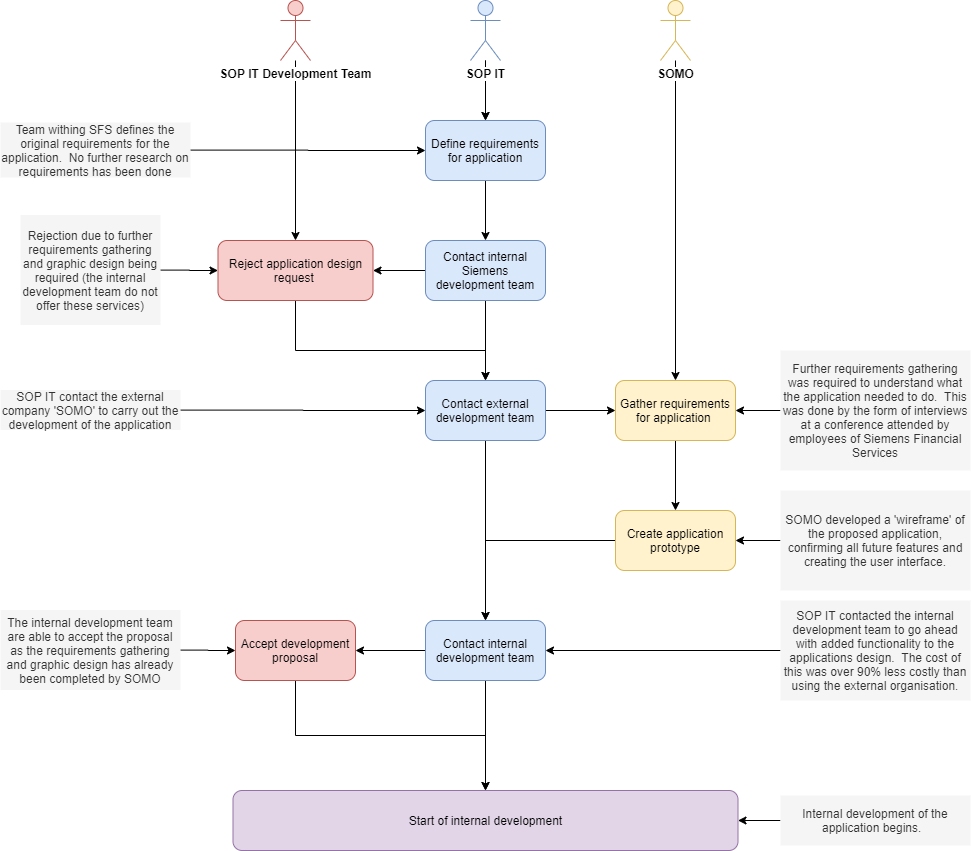


Figure 1.1: Base requirements for application.

The team contacted an internal development team at this stage; unfortunately, more design and requirements gathering were needed than could be afforded by the internal party. As a result, SFS outsourced the initial design and requirements phase to an external company, SOMO. SOMO created a proof of concept for the application, which was showcased to SFS staff and executive management (*see appendix 1*). A group of individuals were selected to act as a user testing group. This selection was intentionally diverse in order to gauge opinions and collate feedback more representative of future users. Individuals were interviewed at this early stage. The users responded positively and gave helpful feedback on what features they required within this application.

Based on user responses and the designs created by SOMO, the project moved into development. Originally this was to be tackled by SOMO, however, the price quoted was over-budget. As the designs had already been created, the development stage could be executed internally by SOP IT. SFS recorded high level data regarding the application’s interactions and functions. This, along with the designs from SOMO were passed to SOP IT for front and back end development (*see Figure 1.2*).



*Figure 1.2 Overview of project conception and roadmap of progress before internal development.*

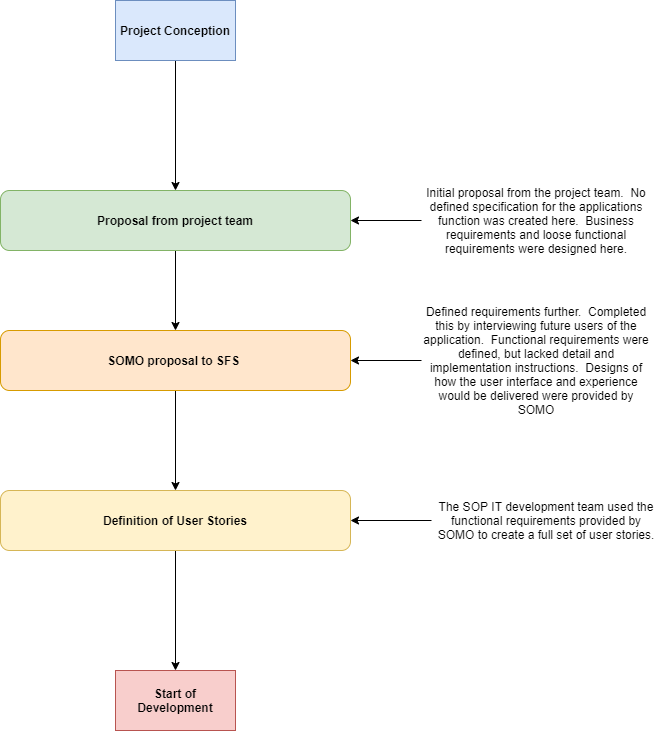
### Overview of My Role

I have been an employee of Siemens since 2018 and was part of the SOP IT team for this project. My role has encompassed many responsibilities including helping to define requirements as well as producing and improving user stories while simultaneously endeavouring to provide overall project management support. I have also been working between the Stakeholder and Developer; helping to pilot test and advance the applications processes.

# 2.0 Application Design

## 2.1 Defining requirements

Throughout this project, there were three steps of defining requirements (*see Figure 2.1*). The requirements phases are explored further below:



***Figure 2.1:*** *Different phases of requirements gathering.*

### **Proposal from project team**

Involved the SFS team presenting to the CFO to get a ‘buy-in’ for the project.  They created business requirements and loose functional requirements.

**Business Requirements**

- Customer First, User driven product

- Established base of existing technologies

- Future ready for further development

**‘Functional Requirements’**

The FFM Application will:

-  Allow Finance Key Control performers to ‘one-click’ confirm completion of activities, delegate to colleagues, promptly log issues, view their personalise dashboards and real time access to support documentation.

- Provide Dashboard views of Key Control completion status for various levels of Management i.e. ARE CFO, Division CFO, BU, Line Manager.

The SFS team had no previous software development experience and therefore required help to further define their requirements.

When the requirements were further defined by SOMO, they were presented to the CFO for approval again.

### SOMO proposal to SFS

SOMO worked to better define what key features were necessary by speaking to prospective users of the application.  They also held consultation sessions with SFS to help further define their overall business requirements (full details of the SOMO proposal given in *Appendix 1).*

### Realisation of User stories

SOP IT worked with SFS to help define technical details and create user stories from the previously defined requirements from SOMO.

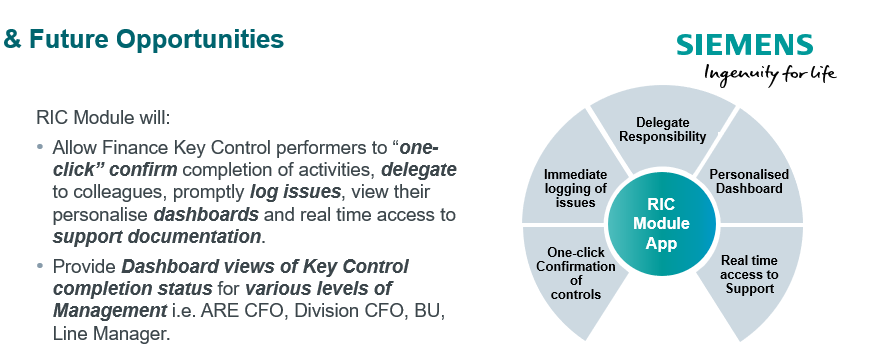
Functional requirements have not been defined traditionally within the project.  They can be likened closely with a stakeholder’s ‘pitch’ to executive management to get approval for an application.   The rough functional requirements that the project team had defined were preceded by the business requirements.

## 2.2 Evolution of Requirements

Below are examples of how a requirement was developed in each stage of the gathering/elicitation.

#### Proposal from project team:

Displays features of the application that were initially proposed in a presentation from the FFM project team to get a ‘buy-in’ from the CFO (*see Figure 2.2*). The requirements displayed, show specific requirements with a lack of detail.

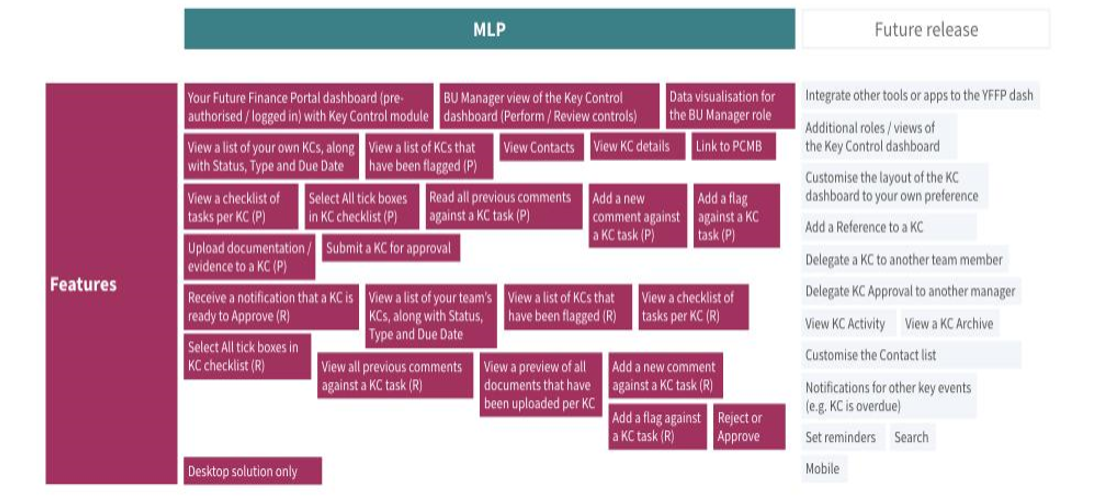


***Figure 2.2****: Displays example of first documented requirements for the application. The requirements can be seen circled in yellow.*

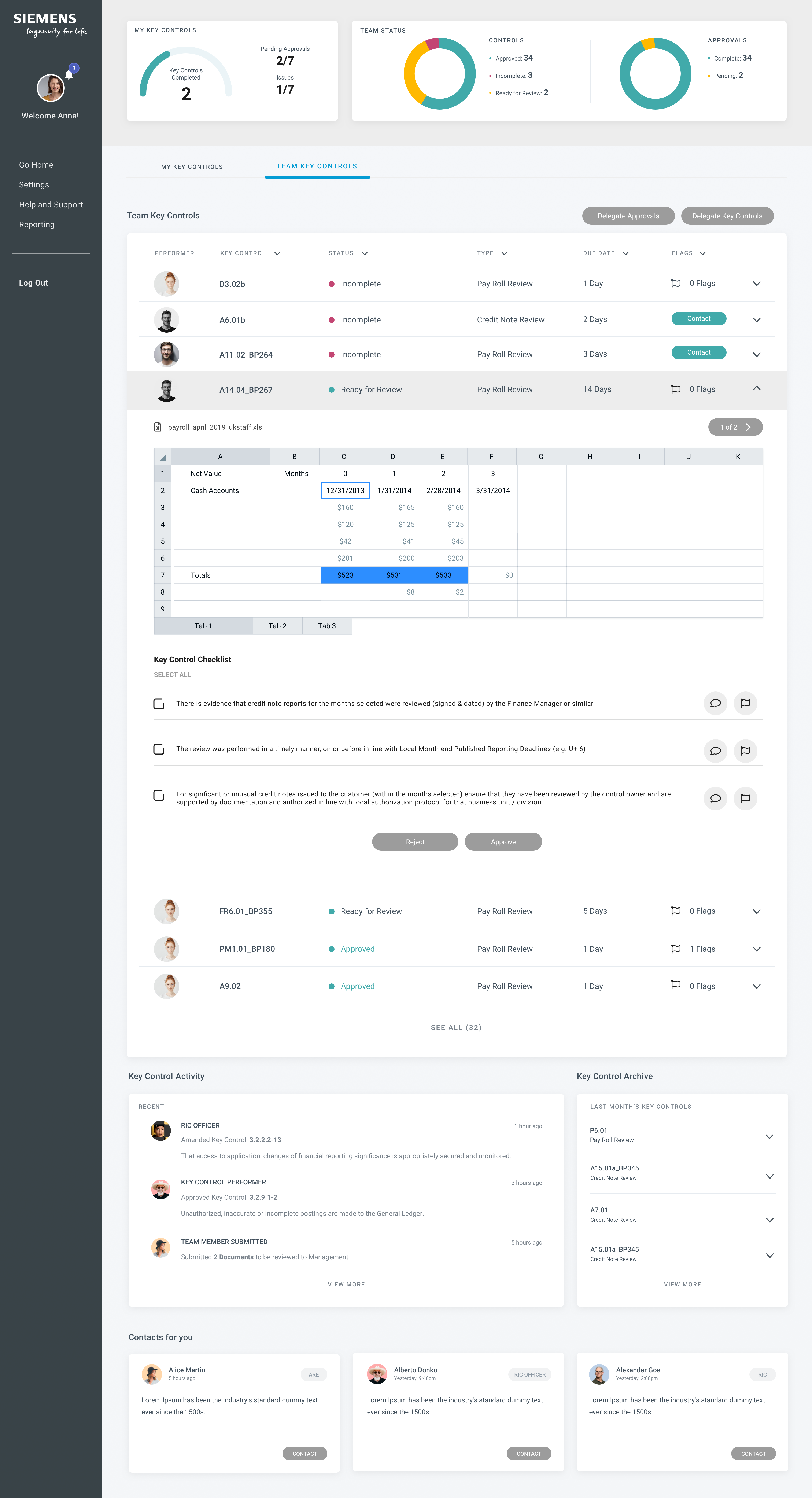
#### SOMO proposal to SFS:

Here the requirements are expanded into separate features, outlining what functions will be included in the application (*see Figure 2.3*). They do not contain the information or criterion that a user story contains, but together make up the MLP (Minimum Lovable Product). The MLP outlines the top features specified by the client, that work seamlessly and look aesthetically pleasing.

Designs for the user interface and user experience were also created (see Figure 2.4).



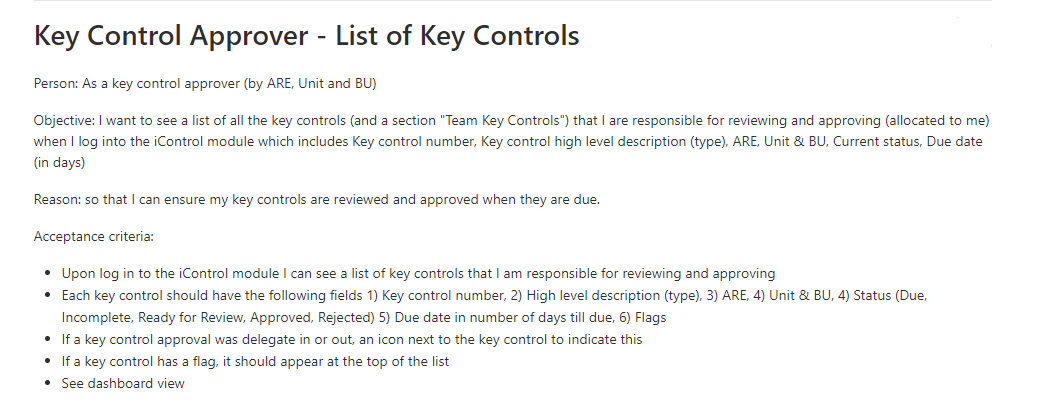
***Figure 2.3:*** *Features list taken from SOMO’s proposal to SOP IT. The features circled in yellow show the evolved requirements from the previous phase.*



***Figure 2.4:*** *Screen design taken from SOMO’s proposal to SOP IT. This image shows how the application would look while displaying dashboard views of Key Controls status.*

#### Realisation of User Stories:

User stories were then defined by the SOP IT development team in collaboration with the stakeholder. For example, some of the features outlined in the MLP have been combined into this user story (*see Figure 2.5*). Further information regarding how this story can be developed and successfully tested is provided within it. This is the last stage of the requirements evolution before development.

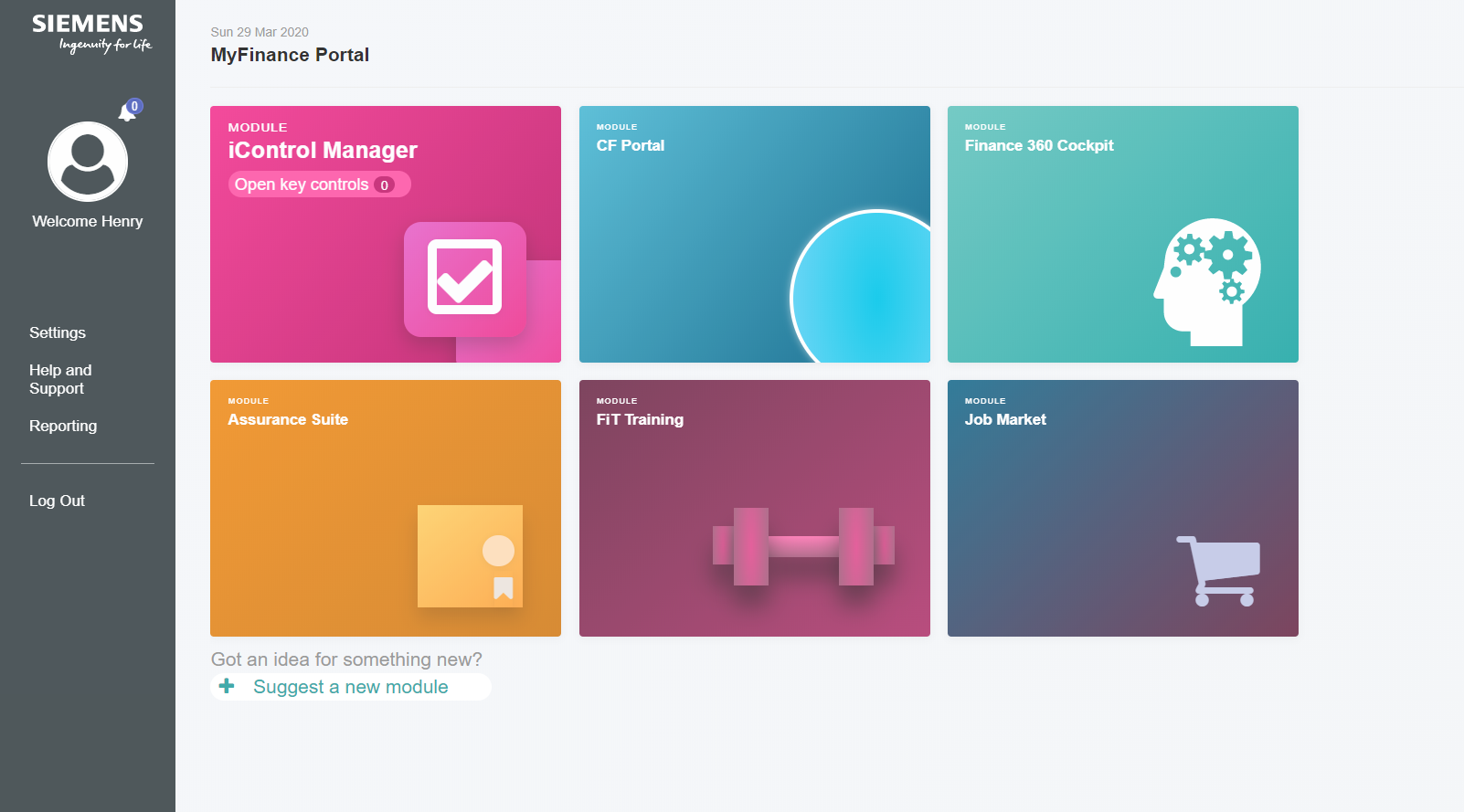


***Figure 2.5:*** *Example of a user story, building from both the features list provided to the development team and the original requirements designed by the project team.*

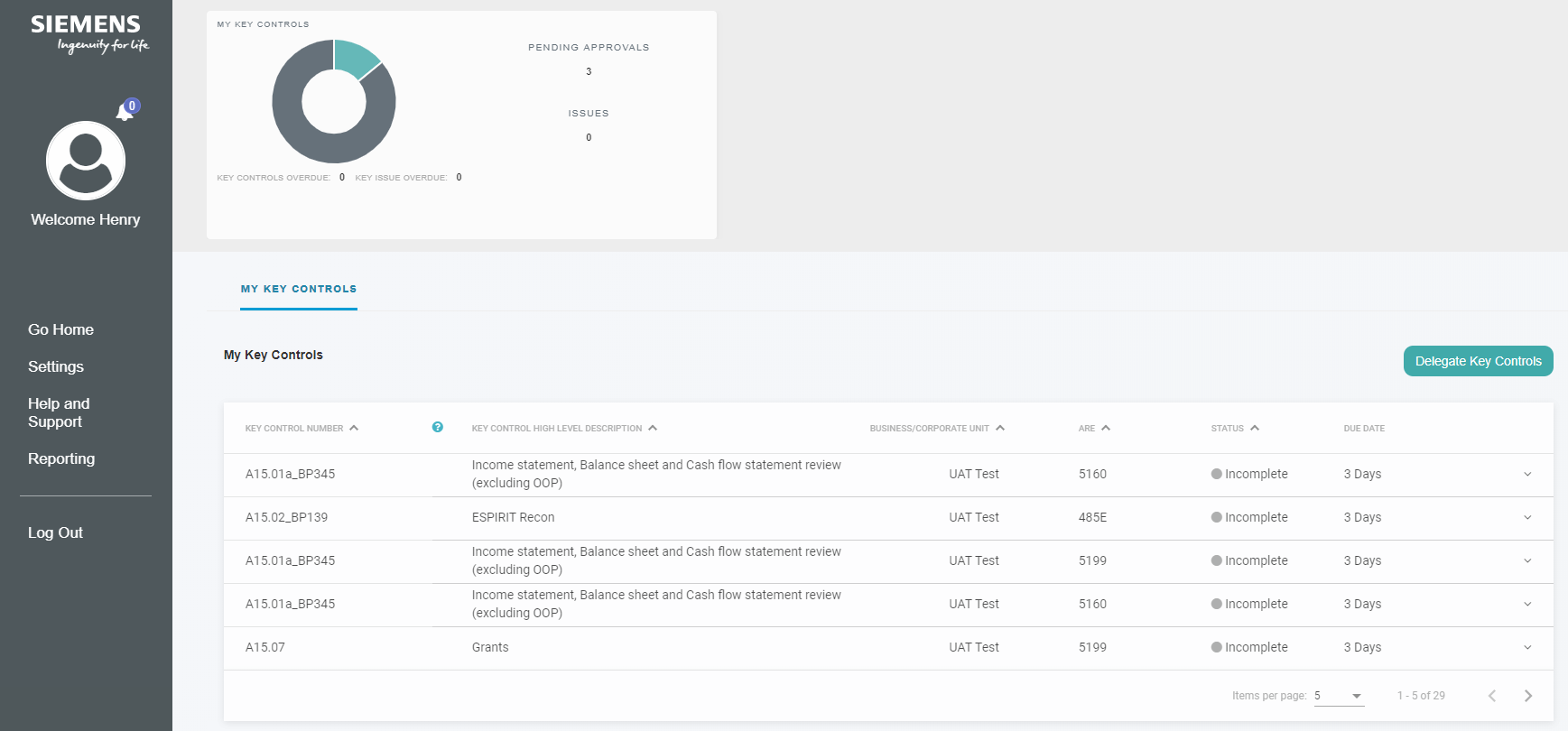
## 2.3 Functional Requirements

The present state of SFS’s working processes is archaic, and considerable portions of the system are manually operated. It was imperative that the application could serve as a catalyst for future SFS users to build upon and should be fully expandable to reflect this. The system was created to host different modules, which aid finance processes. The first was created with the applications launch; being the ‘iControl Manager module’ (see Figure 2.6). This helped departments to regulate key controls, which detect inaccuracies in financial statements. Three major activities that the application facilitates are:

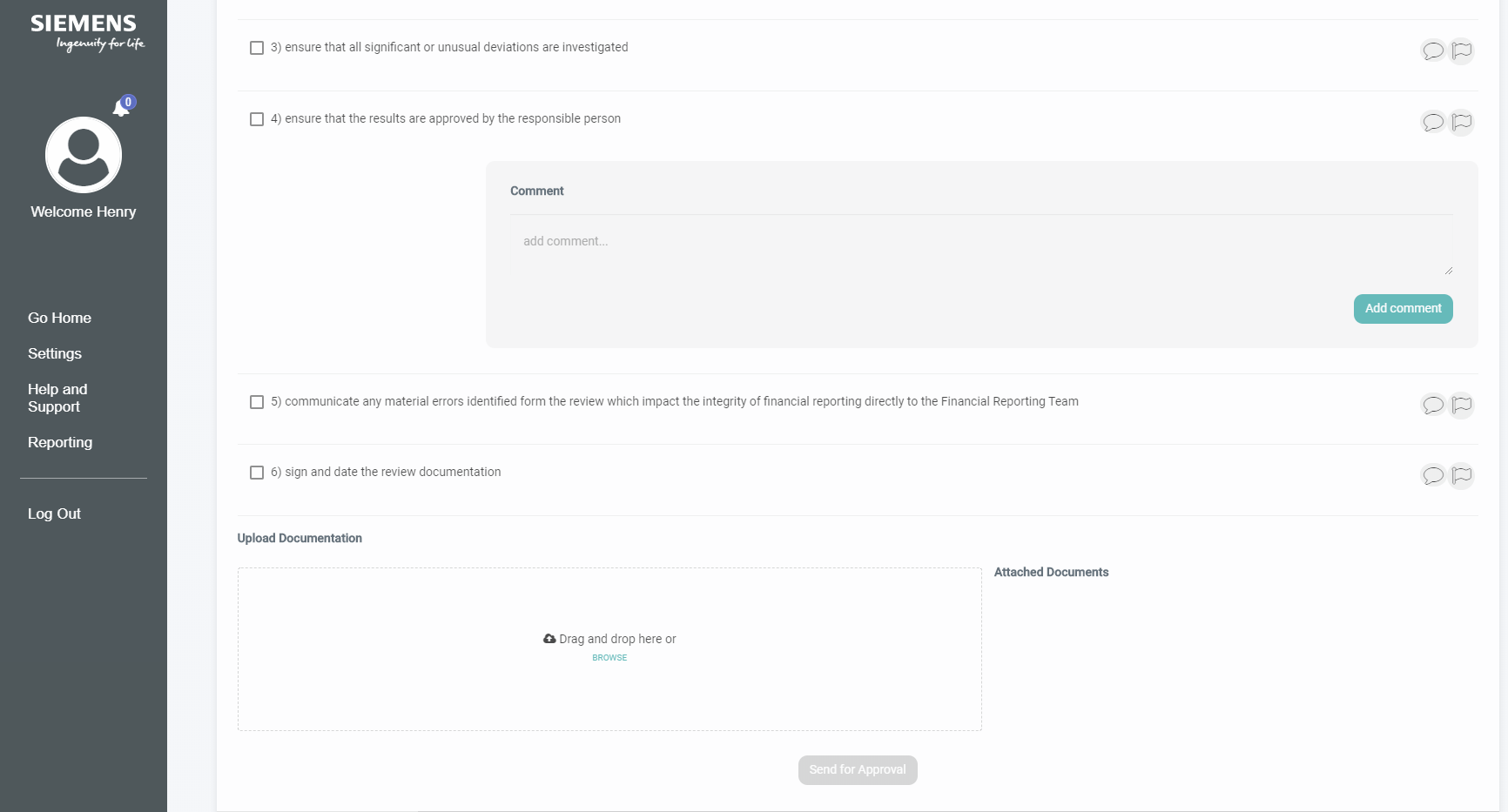
* **Review, approval and oversight of Key Controls (see Figure 2.7).**
* **Gathering of evidence and preparation of Key Controls for sign off (see Figure 2.8).**
* **Assignation and delegation of Key Controls to team members (see Figure 2.9).**



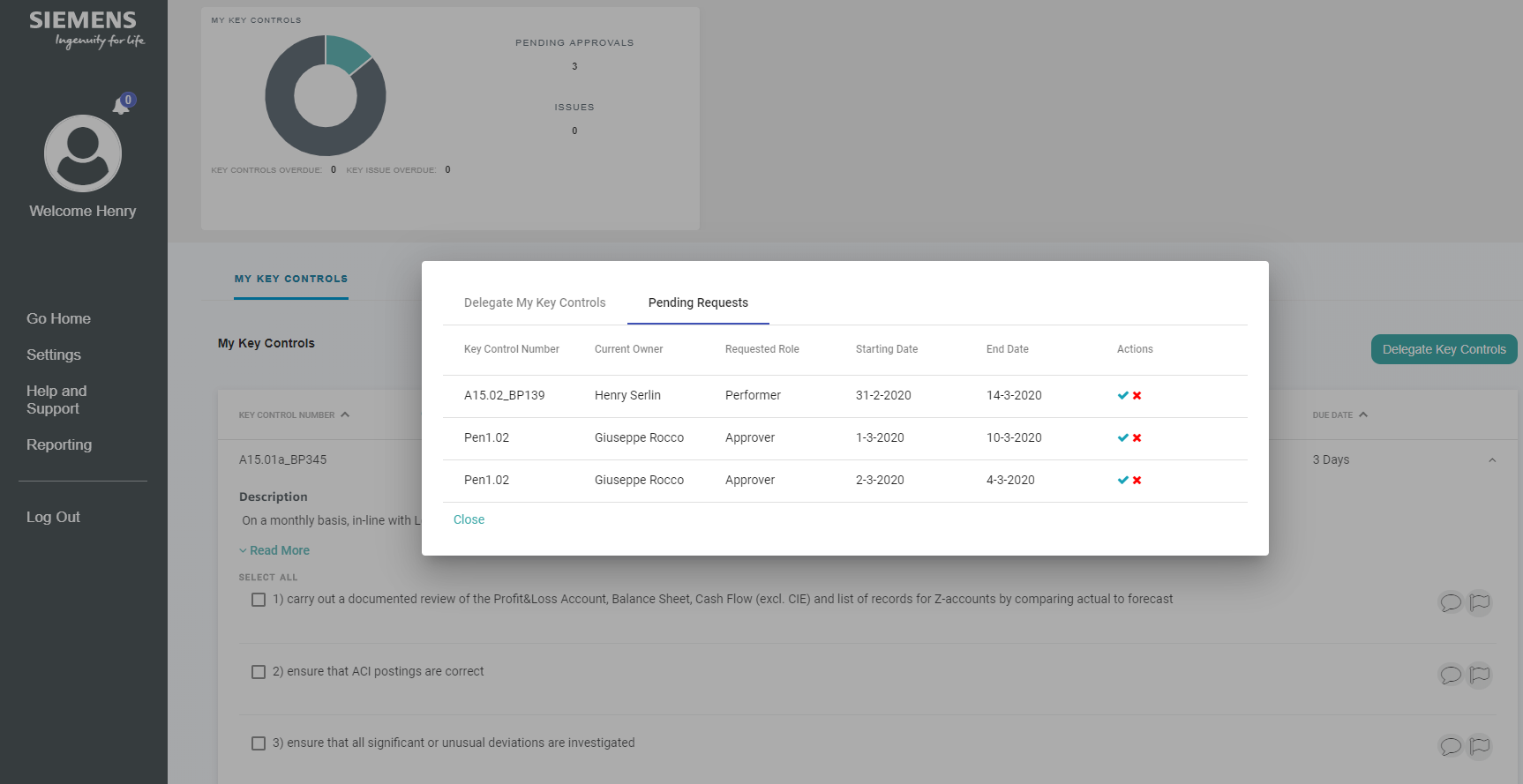
***Figure 2.6:*** *Example of the application dashboard, showing the ‘iControl Manager’, which is the application module that is being created.*



***Figure 2.7:*** *Screenshot of the FFM application, showing the oversight of Key Controls screen.*



***Figure 2.8:*** *Screenshot of the FFM application, showing the evidence gathering and preparation of Key Controls for Sign-off.*



***Figure 2.9:*** *Screenshot of the FFM application, showing the evidence gathering and preparation of Key Controls for Sign-off.*

The projected development of the application was divided into two phases; with Phase 1 being the initial release and Phase 2 being future releases (see Figure 2.10.

|  |  |
| --- | --- |
| Phase 1 | * Allow finance key control performers to ***‘one click confirm’*** completion of activities, delegate to colleagues, promptly***log issues***, view their personalised ***dashboards*** and real time access to ***support documentation***. * Provide ***Dashboard views of Key Control completion status*** for various levels of ***management*** i.e. ARE CFO,   Division CFO, BU and Line Manager. |
| Phase 2 | * **Finance 360’ Cockpit, SFRG’s, FIT Trainings, CFR Dashboards, Datalake, RIC Assurance Suite and Evidence Documentation, Chart of Accounts, Finance Job Market.** |

***Figure 2.10:*** *Requirements phases for FFM application*

## 2.4 Non-Functional Requirements

ISO/IEC 9126 is a standard used to evaluate overall software product quality (International Organisation for Standardisation, 2017). It has been used to evaluate the operation of the application, with examples of each quality criteria provide below:

#### Functionality

* The application (and most Siemens applications) will have a secure entitlement service, in which a user must authenticate themselves by using assigned device and corresponding identification card and password. They can configure this layer of security to allow authorized users to gain access to the system.

As standard, Siemens design applications to work across different approved platforms and devices. In terms of the FFM application, it requires Entitlement Authentication to access the application. This entitlement service is cross-platform and is accessible across any Siemens-approved device.

Other examples of Functionality standards for the application can be taken from the user stories and include:

* When a device does not have a certificate-based card reader, a ‘OneTime-Password’ is configured via a mobile device to authorize the login.
* If unable to access the application, a message should appear to contact the administrators of the site.
* Must display message ‘No GDPR data or sensitive payroll data should be saved’ prior to uploading the file into the portal.

#### Reliability

* Siemens aim to maintain performance across projects. For example, Siemens design their code with this criterion in mind. Best practice is adhered to and they design code to be reliable. Not only this, but they encourage Developers to be forward thinking to anticipate changes and frequently update code. Similarly, they follow design patterns to help solve dilemmas and allow for problems to be revisited and revised easily when required. The FFM application is interface reliant, therefore, the Model View Controller pattern is used to control the development and design of the interface, whilst leaving the rest of the architecture unaffected. This serves to compartmentalise the design, meaning that the entire reliability of the system is not conditional on each of the separate areas functioning correctly.

Other Examples of Reliability Standards for the application can be taken from the user stories and include:

* The system must allow for manual refresh of data. This will happen automatically at the start of each month, however the option for this to be done manually should exist.
* In the event of a refresh, due to a system failure, all in progress transactions should be reset and no data entered by the user should be automatically saved to the system.
* When a system failure occurs, the application should inform the user of what type of error is happening. This error should be logged to be rectified by the admin user.

#### Usability

* They govern the usability of the system through the development methodology. Working with the Stakeholder of the project helps to establish an explicit view of what the user of the application requires. Furthermore, they do usability analysis throughout the project through User Acceptance Testing and Pilot Testing.

Examples of other Non-functional requirements for the application can be taken from the user stories and include:

* + Files must be stored with a standardised naming convention which includes: Key Control Number, ARE, Unit / BU, Month, Financial Year.
  + Notification email should include a link which takes a user directly to the log-on for the Finance Portal.
  + The 'My Reference' column should be minimised (hidden) initially which a user can then expand.

#### Efficiency

* A key example of improving efficiency would be dividing and scaling down the volume of REST calls that transpire throughout the application. For example, when displaying a list of key controls, it presents only 20 (out of 1000s). It displays the key controls that require an action at the top, meaning that a user would not have to request to view all controls to locate one that requires working. Limiting the calls helps to improve the efficiency to the user by increasing the speed of downloading relevant fields.

Other Examples of Efficiency standards for the application can be taken from the user stories and include:

* Display a box which allows a user to either browse and attach a file or 'Drag and Drop' a file, which must allow for multiple files to be uploaded.
* If multiple key controls require notification, only one email should be sent, listing all of those rather than individual emails.
* If a delegation has not been accepted for 10 days, this should be cancelled and removed from the system.

#### Maintainability

* When attempting to solve a problem, Siemens usually work by applying frameworks, which other Developers can easily identify and in turn, edit. Code control and code reviews are also frequently maintained to ensure that code quality is to a high-level, which allows it to be added to and repaired easily; as having it made to a high standard increases the likelihood that others can easily understand it. This project has only one Developer assigned to it, so it does not apply to this.

Other Examples of Maintainability standards for the application can be taken from the user stories and include:

* The references a user adds should be individual to themselves and are saved in the database for future reference.
* The files must be retrievable by the Key Control Performer and Reviewer (I.e. historical files must be available to be accessed).
* The Key control approver should have the ability to re-perform a delegation (temporary or permanent). This should cancel the previous delegation request and remove the flags etc.

#### Portability

* The FFM development environment is run through a Tomcat server, making deploying the environment, and applying edits to it simple. Before the application goes into production, it has 4/5 environments which include user tests, pilot, final and testing; each of these environments are different virtual machines run using AWS. Using an open source cloud service such as this allows for simple migrations between other cloud providers such as Azure or VMware if they required it.

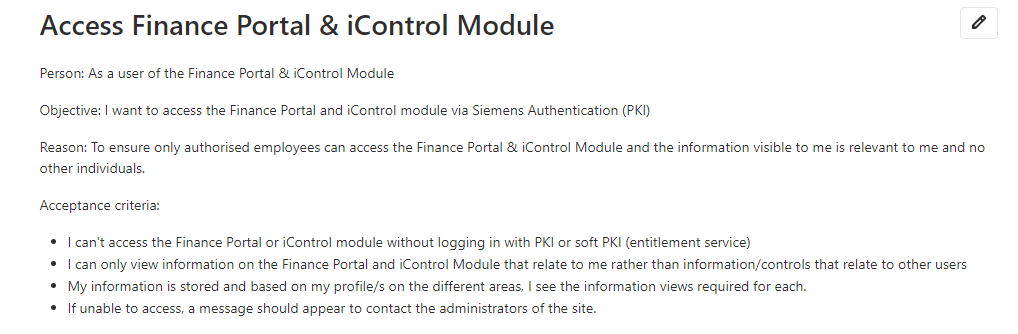
Other Examples of Portability standards for the application can be taken from the user stories and include:

* The iPad view should be optimised to ensure easy navigation and viewing of the applicable screens without the need to zoom in and out.
* The layout defined by the user should be saved and should appear in the same layout when accessing the Finance Portal and iControl module in the future.
* The grey dashboard at the top of the screen (graphs) should be minimised to facilitate the optimisation of the screen when in mobile view.

## 2.5 Overview of Requirements gathering within Siemens

Requirements can be defined from different sources other than a stakeholder’s requirements. Siemens draw many of their requirements from universal company policy. For example, it can be stated in a user story that the application must be secured by ‘PKI encryption’. This is a Siemens standard of encryption and is used throughout the company with every secure application and is expected as a requirement

The User stories that were created for the FFM project contain both functional and non-functional requirements (see Figure 2.11).



***Figure 2.11:*** *Example of a user story containing both Functional and Non-Functional requirements.*

#### Functional Requirements

* A user finance portal has access to the finance Portal and iControl Module via Siemens Authentication (PKI).
* Ensure only authorised employees can access the Finance Portal and iControl Module.
* Ensure information visible to me is relevant to me and no other individuals.

#### Non-Functional Requirements

* If unable to access the application, a message should appear to contact the administrators of the site.
* Information is stored and based on the profile of the user, which is then used to provide different views required for each profile.
* A user cannot access the Finance Portal or iControl module without logging in with a PKI card or soft PKI Pin.

## 2.6 Requirements Traceability

Within this project, the life of a requirement can easily be traced back to. The original business requirements are periodically reiterated by means of an update presentation given to the CFO by the stakeholder, while the functional requirements are kept in a more formal manner. This is done using the Kanban board (see Figure 3.2); where the user stories are stored. All team members have access to the board, where versioning control applies. Meaning that previous requirements within an individual user story can be referred to if necessary.

# 3.0 Project Management Within Siemens

## 3.1 Overall Project Management Within Siemens IT

Siemens IT intends to operate an ‘Agile’ methodology throughout the entire organisation. Employees are encouraged to follow basic Agile principles throughout daily working life and when undertaking projects. Training in the following areas is given to allow this:

#### Kanban

* The Kanban method is an Agile-based framework aimed to aid the completion of a high number of varying tasks (Rehkopf, 2020). This is a non-project specific method, and as such is taught to be used on a personal basis, with the intention that it can be applied to whole projects. When used personally, ‘buckets’ are employed to determine the stage of completeness of tasks, which are then ordered based on priority and individual capacity. The private use of these boards breaks Leopold’s first principle of Kanban usage; in that a user should focus on ‘Making the Work Visible’ (Klaus, 2015), as Siemens encourage the private usage of these boards. Subsequently, the primary focus of Siemens Kanban use is not to ‘improve collaboratively’. It is to follow Emerson’s personal Kanban style and to visualise and limit work currently in progress (Emerson, 2020).

#### Regular meetings

* The second Agile principle taught to employees is to meet regularly and discuss project progress and effectiveness as often as possible. It is understood that due to the size and disparity of work within the organisation, having a regulated formula for Sprints would not work effectively. This is applicable across the company not just in a software development (SD) setting. The considerable size of the organisation means that working actions and approvals can take longer to complete due to regulations and processes in place; which can develop into longer lead times for a project.

Siemens reflects the view that understanding requirements around sequential iterations increase the likelihood that earlier decisions may have to be revisited, leading to difficult or costly operations (Lonergan, 2019). These general working practices use a more flexible variation of a ‘Sprint’. There is no formalised process or goal to meet each week, but each ‘Sprint’ is used as a chance to catch-up with work and to alter any project specifications.

#### Providing customer satisfaction through continuous delivery

* The third principle is a focus on providing customer satisfaction through continuous delivery; which supports the ideology of the internal-facing IT division. This builds on the SD practice of frequently deploying incremental software changes to a shared production environment, e.g. GitHub. This is done to help increase customer satisfaction through increased involvement in the development process and reducing time and costs spent through continuous testing. Outside of SD, Siemens continues to encourage this working method by updating, involving and working with customers closely.

#### Critical Reflection

After working within the Siemens environment, I would fully support their methodology and take on Agile. The literacy surrounding Agile methodologies is written primarily concerning a specific project, while Agile within Siemens is taken more wholly as a general working atmosphere. Having experienced this first-hand, I have found it works most efficiently when it is employee driven. The aforementioned aspects of Agile can all be used individually; however, they work most effectively when implemented as a team.

The lack of a set structure can however lead to employees not adopting this working style, hindering progress in a project. Within Siemens there is flexibility for project managers to decide on whichever Agile framework to use with a team. This can help encourage adoption among those who struggle to follow all the Agile practices.

## 3.2 Project Management for Software Development

Internal SD conducted by Siemens is decided on a case-by-case basis when concerning Agile frameworks. Although general IT operates by loosely following the Agile methodology, Software Development Teams (SDT) will usually adopt a fuller process, following the methodology much closer. When an SDT receives a project request, the manager of the team will assess the scale of the project to decide the size of the team that will be required.

A Team Lead will always be delegated to a project, which in most cases is the same manager that is assessing the project size. Their role is defined chiefly by them managing the project. The manager then assesses the need for an Agile framework to be implemented into the project. They can choose a framework based on many factors, the primary being the length of development and manpower required. However, a vast part of the decision towards which framework is chosen is based on the people that are working on the project, considering their comfortability and efficiency in working with it.

#### Combining frameworks

There is room to allow for different aspects from various Agile frameworks to be used collectively when the extent of the project is relatively limited; therefore, requiring flexibility of work style. An example of mixing ideas from frameworks is ‘Scrumban’, which uses the Sprints of scrum, while maintaining the ‘work in progress’ and incremental change tracking of Kanban (Pahuja, 2015).

It is important when combing multiple frameworks that the core elements of each stay intact. Furthermore, it is key the team sees an overall net-gain from time spent using the framework (Stellman, 2019). The framework can change over the course of a project.For example, during a large-scale Scrum project, where a considerably sized team is used, if the scope of project were to get smaller, then the Kanban is adopted; and vice versa.

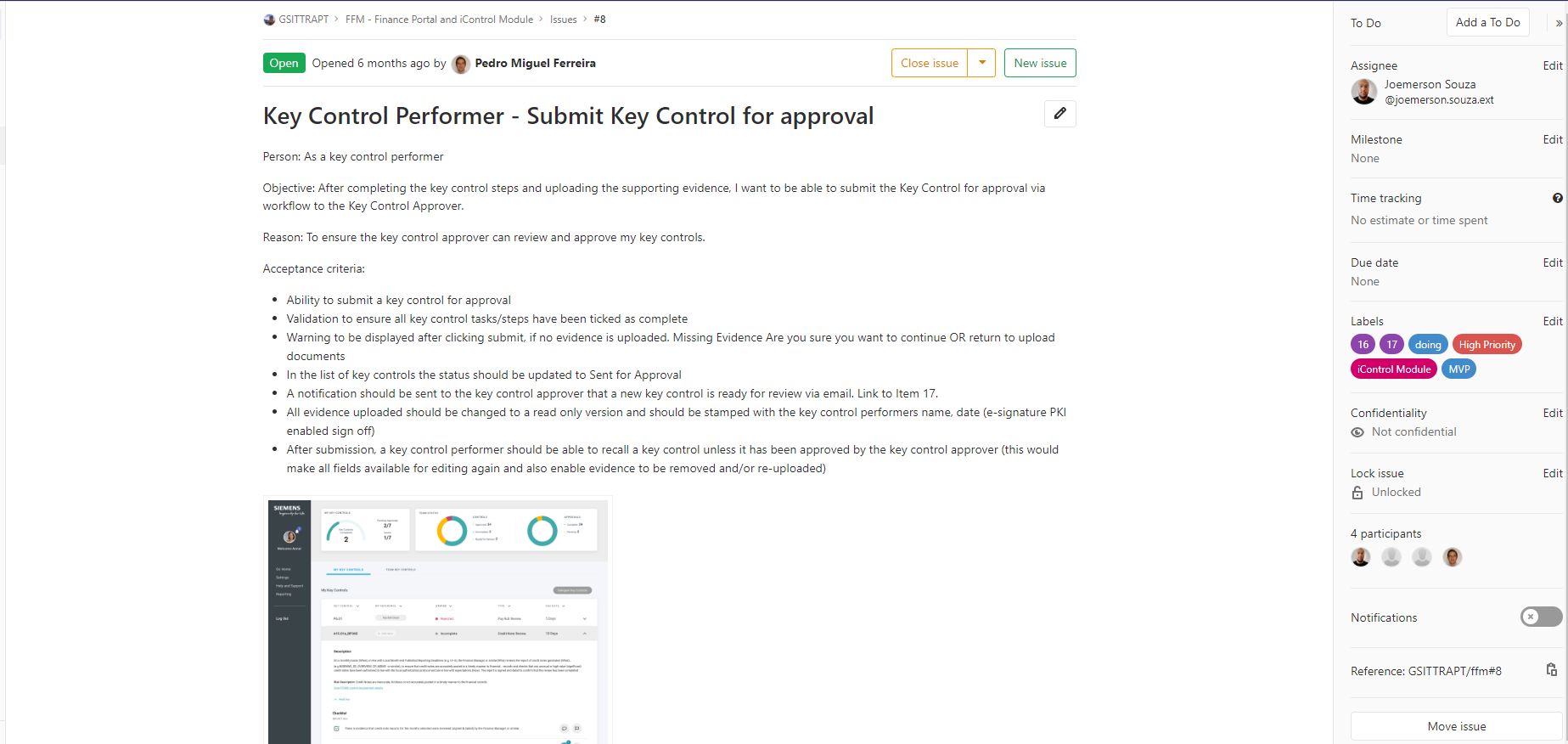
## 3.3 Project Management for the Finance Application

The finance application being created was not sizable enough to have a specific framework applied to it. As previously mentioned, this allowed for a combination of techniques to be united. The project has been run in close collaboration with the customer, having them feature in the project team under the supervision of myself to create a pseudo-product owner role. This could be managed because of the internal client; allowing more of their time to commit to the project; which adheres to the classical Agile working theorem.

#### Commencing a Project and Defining User Stories

The development team and stakeholder defined a large list of user stories in an initial meeting User stories are defined between the PM and the Stakeholder initially. When the PM is comfortable that the Stakeholder can correctly create user stories, then the Stakeholder will take over in creating these.

Before beginning development, the PM will approve the full list of initial stories. This is done so the extent of development is clearly agreed upon. They use a standard method for creating these, which is the ‘As a (user), I want (goal), So that (reason)’ definition process. Siemens follows this protocol for defining user stories, but encourage more details to be included, such as acceptance criteria, scenarios and mock-ups when available (See Figure 3.1).Denys (2016) would argue that the user stories created by Siemens were closer to ‘Epics’ as ‘they are a big piece of product functionality’ and mostly ‘too big to be completed in one Sprint’. However, I would disagree with this somewhat by arguing that the user stories are not all of appropriate size to be classed as this. Instead, there are a mixture of epics, user stories and longer user stories; all of which remain defined as ‘User Stories’ by Siemens.



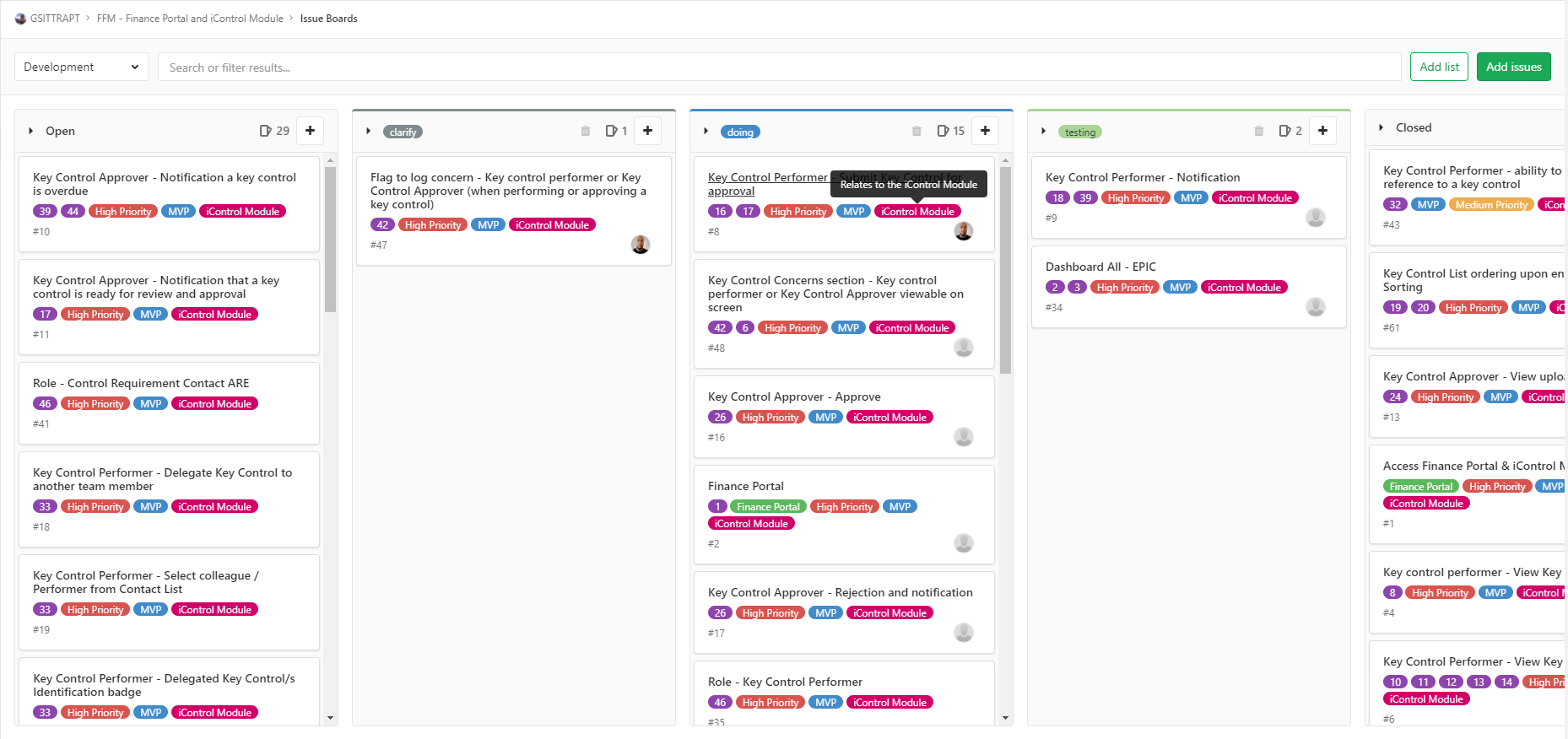
***Figure 3.1:*** *Example of initial FFM user story, showing User, Objective, Reason and Acceptance criteria, along with priority and example images.*

If a Stakeholder doesn’t have the full time to commit to the project, they are encouraged to cancel it or hand over control to another party. There is great importance in the relative level of communication and work produced by Developer and Stakeholder alike. This echoes the statement that ‘*If either side dominates these communications, the project loses*.’ (Cohn, 2004). The Stakeholder defines the business logic, so it is their responsibility to define, refine and answer questions about the requirements they are requesting.

To improve start-up time and avoid delays resulting from a lack of understanding by the Stakeholders, Siemens have started an 'Agile training Coaches’ initiative. This consists of a team to help others to adopt Agile approaches into their projects. One aim of this initiative is to teach how to correctly write user stories, prioritise work and correctly maintain a product backlog. Personally, I found that colleagues having equivalent training to myself to be fundamental in undergoing an Agile project as everyone worked in the same manner.

#### Maintaining User Stories and Using Kanban

When the list of user stories was defined, they were uploaded to GitLab and added to a ‘Kanban’- style board. The board defines the different buckets that are used to symbolise the state that the user story is in e.g. *‘defined, requires clarification, working-on, testing and completed’* (see Figure 3.2). Both the Stakeholder and the Developer would have access to these user stories, with the Developer and PM moving them into their respective buckets when required. For example, the Developer could move a user story into the ‘clarify’ bucket if they needed more advice regarding how they may complete it. They would also move a user story into the ‘working-on’ bucket when developing the story. They would then move it to the ‘testing’ bucket when it is ready for the client to test functionality. The PM then has access to ‘complete’ or create a new user story. The Kanban board used by SOP IT is in the form of an Issue Tracker, that is hosted within their GitLab environment. Although it is intended for issue tracking, it has the same functionality as a Kanban board and is used here in the same way.



***Figure 3.2:*** *FFM**project Kanban board provided through Gitlab, showing both buckets and user stories within them.*

As well as unarranged update calls, weekly meetings are held between all members of the project team and the Stakeholder. These calls are held to discuss progress made on the development of a story, raise queries and to add or remove a story. They document updates to testing progress on GitLab by use of comments on the user stories. For example, the Stakeholder providing feedback on what works from a user story. During this they would highlight what is still not functional or working as it should be. Commenting on user stories is not exclusive to testing and is used to update a user story with new information. The commenting system is chronologically stored, so if many tests are required on a single story, they can view them can all with the most recent appearing first.

## 3.3 My Role

At the start of the project, my main role was to work with the Stakeholders to better clarify user stories that had been created. As the Stakeholders lacked experience in creating user stories, it was difficult for the Developer to understand and accurately produce what was necessary. By re-writing the pre-existing stories, I was able to provide more clarity, and this allowed the project to move forward more rapidly.

As the project progressed, I worked with the Stakeholder more closely, educating them on how user stories can be made to be clear and gain maximum understanding from the Developer. The initial backlog of stories was very large, and it soon became apparent that the completion of the stories was not workable within the time budget agreed. I worked with the Stakeholder to prioritise and condense the stories that enabled the application to be fully functional. My role continued in this way for the duration of the project, working between the Stakeholder and Developer to create, edit and clarify requirements. From a development perspective, I was able to translate constraints and issues from the Developer back to the Stakeholders. This feedback would then be used to alter any stories that required it.

# 4.0 Agile Working Methods

## 4.1 Level of Documentation

The Agile Manifesto for SD states that ‘Working Software over comprehensive documentation’ is prioritised (Beck *et al*, 2001). This is to prevent development teams being ‘bogged down’ by trying to complete documentation. However, it does not mean that documentation should be written off completely, just not prioritised (Eby, 2016). The statement ‘*a document can be considered to follow Agile principles if it is valuable, essential, and created or updated just-in-time’* supports this idea(Smartics, 2015). To further build upon this, a document can be considered valuable if the Stakeholder can gain useful information from it. It is essential if it provides the Stakeholder with key information and it is just in time, meaning that a document is created at the point in time when it is required and not beforehand (Linders, 2014).

#### Siemens Approach to Documentation

Within Siemens, documentation is broken down into two types; technical and end-user. Technical documentation refers to the functionality and approach of the product being created and is mostly the Developer’s and tester’s responsibility. There is no standard for documentation required to be produced, however, the use of applications such as ‘Swagger’ which document API structure are encouraged. Technical documentation also can refer to a wiki that is maintained by the development team. When a project is being run using the Scrum approach, the Scrum Master is the one responsible to remind the development team that they require documentation to be created.

End-user documentation, however, can refer to high-level project management documents, external test-cases and guiding documents. This documentation is completed by the client in most cases. As most development is done internally facing, this is not to be confused with technical documentation.

#### Documentation for FFM and its relevance

For the FFM application, Siemens manage much of the documentation that they use by way of their user story and commenting system. However, as this encompasses most of the documentation that is created, the overall level of documentation could be considered to be too sparse. However, it is adequate for bridging a gap between Developer and Stakeholder maintaining consistency in development (Selic, 2009). Although the documentation is not the direct focus when using an Agile methodology, it should not be disregarded entirely, nor completed extensively. Dependent on the size of the project being undertaken, it should alter the level of documentation. Moreira writes that *‘The level of effort applied to write and maintain the documentation plus the value of that written document should have a greater return on investment’* (Moreira, 2013). This reflects the current process that Siemens uses for this project. As this project was relatively small in terms of scope and manpower, it did not require additional documentation to be completed. Outside of the FFM project specifically, Siemens still emphasises interaction over documentation where necessary.

They do not document Software designs, with the Developer using the user stories to plan and track further development. Siemens concentrate heavily on working to deliver code. This reflects the stance that *‘the cost of producing documentation when the same time and resources could be used to generate more code’ (*Selic, 2009).

This has clear advantages for both Stakeholder and Developer. The Stakeholder can benefit from the increased contact and involvement in the project. Doing this proves to be useful when trying to implement a specific idea or add additional functionality to a system through increased understanding and clarity. Meanwhile, a Developer can concentrate on the development of the code, without dedicating a large amount of their time producing documentation.

The PM is among those who experience disadvantages of working in this way. The PM may find it difficult to gauge a sense of where the project is at and may also have difficulties in future planning and reflection. They may struggle with this as the development process has not been as thoroughly recorded.

#### Critical Reflection

I think it is important to prioritise getting work completed and delivered to a user, rather than maintaining documentation. When a project is of a small size (such as FFM), a large amount of working time can be taken up creating detailed documentation. As the Stakeholders were not experienced in requirements gathering/analysis, the specification to the Developer changed frequently, creating documentation for the project would not have been useful. This project also had rapidly changing requirements or developments and as such having updated documentation would have been incredibly time consuming. Instead, creating a product that the Stakeholder could use and develop their requirements further from is the best approach.

## 4.2 Sprint Cycling/Working Practice

#### Overview

For the FFM project, the Agile Alliance would describe this method of working as the ‘three amigos’ method. Where work is examined by three different perspectives from before, during and after development (Dinwiddie, 2014). In this method, each person who holds knowledge about these areas collaborate to share understanding about each area of work. This method helps to identify any confusion at each stage of the project early and encourages discussion between team members that help with the functionality of a product.

#### Sprint cycling within Siemens

When undergoing a project that implements Scrum and uses Sprints, Siemens adopt the approach that ‘Agile frameworks should be adapted to the team and not the other way around’ (Ferreira, 2020). The Sprints are more likely to be upheld and done with tighter regulation when a project is more complex; which equates to it having a larger team (which is viewed as having between 6 and 10 members). Ultimately, this is something that the Scrum master decides and conducts.

#### Critical Reflection

The working practices applied to the FFM project proved effective and we delivered great results; as seen by the feedback from users after the pilot test. However, halfway through the project, the sole Developer was changed. This meant that as they had not worked on the project, they were not accustomed to the team’s way of working. This resulted in a period of transition preventing the Stakeholder from addressing questions with them each. Although largely unavoidable, to improve a situation like this, the Stakeholder could have explanatory sessions with the new Developer and the project manager. Despite each employee within SOP IT being given Agile training, it is dependent on the individual how they apply it. This means that its effectiveness is variable, considering some employees may utilise it in disparate levels of competence.

# 5.0 Testing

## 5.1 Application testing within Siemens

#### User Acceptance Testing

User Acceptance Testing (UAT) is carried out throughout the development of the application, working on ensuring each user story that is developed functions duly. Cimperman (2006) writes that UAT’s purpose is to ‘*validate that a system of products is of sufficient quality to be accepted by the users’*. This echo’s Siemen’s values, but is performed differently to the traditional testing method. UAT is used to provide a binary answer to requirements, from which either they can create further tests or user stories, or they can pass a user story. The requirements to pass a UAT test are defined within the User Story but can be divided into sub-requirements through this testing. For example, if a UAT highlights a minor flaw not sufficiently large enough to create a new User Story, then it will be added to the commenting section of the User Story (*s*ee figures 5.1, 5.2), defining a new sub-requirement. The UAT testing is undertaken by the Stakeholder, as they will be the end-user of the system.

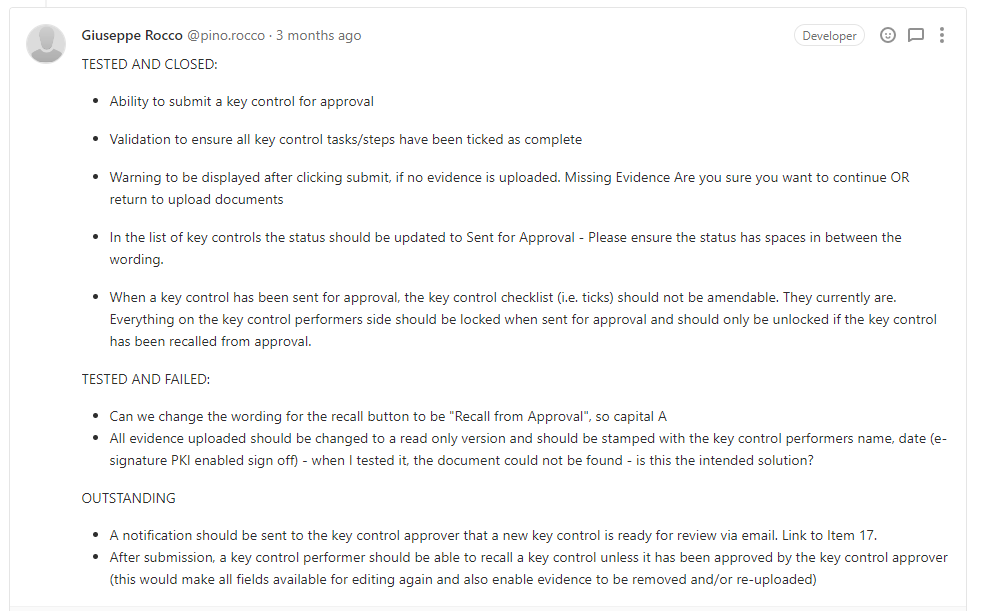
The commenting system works by the stakeholder commenting on what they have tested on the user story. After the tests are run, the stakeholder then comments to confirm what is working as it should be (defined within the user story) and what needs more development time. The developer can then reply to the test comments to confirm that they have been reviewed, so that the stakeholder is once again able to test them. The developer may also comment on these tests, by explaining their working in the case that the stakeholder is using the application wrong.

Compared to more traditional development methodologies; such as Waterfall, this shifts the focus of a tester being solely responsible for identifying threats at the end of a project, to a tester being more responsible in preventing them mid-way through development (Collins, E., Lucena, V, 2012).

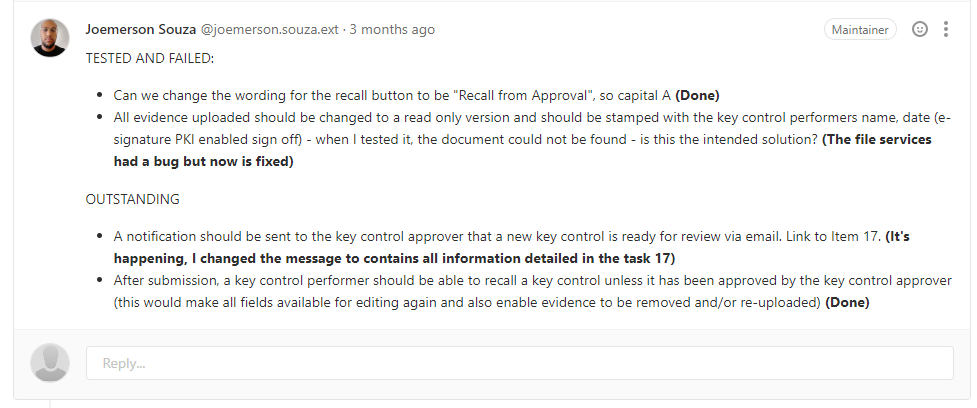
Lykourentzou *et al.* proposes the use of updated wikis to support UAT within an Agile setting; as they can be used collaboratively, meaning they can share testing ideas amongst the test group (Dagka et al, 2010). Siemens expands on this, through the aforementioned Kanban documentation process for testing; which because of its design, allows for a more fluid and accessible approach than a wiki.

UAT is performed rigorously in Agile and more frequently, as there is a constant requirement to deliver a working product to the Stakeholder. Such as displayed in the FFM project, Agile encourages UAT throughout the development of user stories through to the end of the development. If this project was being conducted through SCRUM by using Sprints, there would be more than one layer of UAT testing. This would involve a ‘Sprint review’ at the end of a Sprint where the wider team would run through what has been created to receive feedback from a wider peer group, whilst ensuring that work between the Sprints is not hampered by the meeting time. (Macleod, 2020).

A final Sprint would be set out in order to wholly conduct UAT testing after development has been completed. This is done to test the entire system as a whole, rather than smaller parts of the system as tested previously (Cagley, 2013). Siemens bypass the two extra levels of testing. They prefer to test the whole system regularly, ensuring that if a fault is found, they find it early, rather than at the end of development.



***Figure 5.1:*** *Example of ‘commenting’ system for a user story. Displayed here is the continuous UAT testing from the Stakeholder. The completed and function points as well as the outstanding and failed points are documented within the individual user story.*



***Figure 5.2:*** *Example of ‘comment reply’ system for a user story. Displayed here is replies to the Stakeholder UAT comments. For simple tasks, the Developer can communicate back to the Stakeholder to further progress testing and answer questions.*

#### Pilot Testing

After they have created the basic system; meaning that they have successfully implemented the base functionality required for users to undergo work, Pilot testing will begin. To do this, they choose a varied group of future users to test the software under normal working circumstances. They will work as they would normally, on a running version of the application. This is done to receive feedback about how the application can be improved; specifically looking at finding any issues in functionality or usability. The purpose of this testing is not to have the group of users use the software permanently. Instead, it intends for them use it for a specified period e.g. two weeks, so they can use it in a variety of normal working situations.

After the testing period is over, they will gather feedback to adjust the application. They take feedback for this through a pre-defined form, with questions that cover the key business process steps. They make the feedback forms intending to gather information on how the current system could be improved, not the processes that the system is based around. Questions asked are open-ended and encourage the Pilot testers to provide recommendations for improvements, rather than suggestions of what worked well. When the feedback is fully received, it will be reviewed, and the key points will be extracted. The points can be evaluated to either create new user stories, or added to existing ones, or if they can be quickly implemented will be changed after a quick discussion with the Developer. Testing in this way requires the Developer to create a ‘pilot environment’, which is separate from both the UAT and Testing environments; this allows for the users to freely use it.

Before Pilot testing begins, they require several tutorial sessions to be held. These are conducted as they may receive otherwise incorrect feedback based on areas of the application that are not yet fully functional. It is important to do this to ensure pilot users provide a relevant assessment of the functionality.

Pilot testing rarely specifies defined requirements and is intended to allow real-life users to freely use a defined part of a system. This can bring about a multitude of issues. The timeframe for correct testing being one. The end goal of the pilot testing is to get a comprehensive review of how the product works under everyday usage. Everyday usage of a system does not involve using all features that are possible. As a result, it can take a long time for a user to use this ‘naturally’ and without prompt.

Beta testing can be used as an alternative to pilot testing. Typically, this would be done on a larger-scale project wherein the application is perceived to be ready for final release by the development team; as its purpose is to find final problems and defects (Gupta, 2016). The primary difference between beta and pilot testing is that beta is performed with the entire user group, rather than just a selected few. As Beta involves a larger user group, it will be a true reflection of stress, with more performance-related issues being able to come to light. More users help to better highlight key issues. When undergoing a pilot test, it is difficult to see which issues are individual to a single user or are common among the entire user base. (Ventayol, 2016). However, as the users for the Beta test are not individually selected for suitability, valid feedback and understanding of the application cannot be guaranteed.

#### Testing during development

From a development standpoint, testing is very limited and is heavily scoped. All that is done are unit tests, which are intended to be atomic-level tests for a singular algorithm or method. To a degree, Developers also test what they are creating overall. However, this testing is somewhat biased by their individual understanding of the requirement and by the individual way that they have been testing its functionality (which can be limited).

#### Unit Testing

Within larger-scale projects, unit testing is commonplace within development. As much of the framework for the FFM application previously existed, time spent unit testing would be unnecessary, especially as many of its components already proved to be functionally sound. As the project was small-scale and had a limited development time it was important that time was utilised correctly, and the focus was on ensuring a working product was delivered to the stakeholder. This meant time was prioritised to creating functionality, rather than correctness. As the resources for User testing were much greater than in-development testing, any faults that would cause issue with the function were expected to be revealed in this manner, rather than by unit testing.

#### Critical Reflection

I found that UAT was very useful for all within the team. It enabled the Stakeholder to gain a better knowledge of what the Developer could do and how they could enable them to understand the requirements better. It also helped the Developer to solve the Stakeholder’s requirements more accurately, as the testing cycle had a fast turnaround. To improve efficiency of this testing, I would suggest that they would set a notification system up between the Stakeholder and the Developer. Doing this would allow both parties to be fully aware of when they had made changes or updates and allow them to act upon them quicker.

The pilot testing that was carried was very beneficial and brought issues to light that otherwise would not have been realised. Providing enough training to the users beforehand was key to being able to gather useful feedback. For future Pilot Tests, highlighting the constraints of the provided system would be advantageous. This would ensure that there is not feedback gathered that relates to issues that are out of scope and cannot be amended.

## 5.2 My Role

I was involved throughout the process of UAT testing. Most of the application’s functions required two users interacting with each other on the platform. I took part in several meetings weekly with the Stakeholder to performs tests on the system. We operated these tests all within the bounds of a user story that had been moved to the ‘testing’ bucket in the Kanban board. I would then help to evaluate if a test was fully completed, document failures in the tests, create new tests dependent on outcomes and articulate responses back to the Developer. We would communicate with the Developer whenever these tests were completed and regularly check the Kanban board to see if there was any new functionality to be tested. Weekly, myself and the Stakeholder would meet with the Developer to discuss the week’s progress. During these meetings we would discuss the following week and evaluate the overall progress of backlog of stories.

Once the base functionality of the application was developed, we decided that pilot testing should be carried out. To do this, I worked with the Stakeholder to compile a list of relevant, open-ended questions, which were aimed at testing the functionality of each area of the application. The pilot testing group was chosen by selecting a varied group of users who would have access to the application upon its release. I was able to help create training videos to show the pilot users how to properly use the application. We then gave them two weeks to use the system; at the end of which I provided them with a survey to complete. I compiled the results of the survey and highlighted the key relevant points that could improve the application and then could add them to existing user stories.

# 6.0 Conclusion

In my opinion, the FFM application was developed efficiently and to a high standard. This was largely due to Siemens unique use of the Agile framework. Most notably, the constant contact upheld between the stakeholder and development team.

The adoption and adaptation of Agile has been achieved through the compulsory training of employees, allowing them to apply its principles to everyday working. The flexibility of Siemens’ approach, along with this training has allowed employees to utilise differing principles according to the scope and style of a project. This undoubtably allows for increased confidence, and therefore comfort, in workflow processes maximising the quality of work whilst effectively managing time.

In terms of the FFM application, the stakeholder, being an internal party allowed for increase input and communication. This increased communication provided an increased understanding in the changing statuses of development. For example, adhering to creating user stories in the standard way, but expanding the scope of some to be closer format to ‘Epics’. As both parties consisted of Siemens employees, the Agile principles were adapted into this project easily. Had this project been completed with either an external stakeholder, this project would not have been so effective.

Siemens could benefit from the adoption of a specific software to host Kanban board. Currently, an issue tracking tool is used to host this. However, problems may arise when issues need to be tracked separately from the Kanban board. This could be avoided if specific software is used to host the board.

Furthermore, Siemens stand to benefit from having a greater understanding of functional and non-functional requirements. They are currently overlooked and mixed together within user stories. If these were split and adequately numbered, it would allow for better tracking during testing. Having a deeper understanding of non-functional requirements would be advantageous and allow them to be measured against a quality standard such as ISO/IEC 9126, improving the delivery of products that are created. Understanding functional requirements provides a clearer view of the ‘must haves’ in the software, which in turn helps to prioritise what the development team should focus on.

Using this project as a case study, it can be derived that the FFM project being successful is a reflection on Siemens’ fruitful adoption of Agile practices.

# 7.0 References

Ferreira, P. (2020) Email to Henry Serlin, 09 March.

Cohn, M (2004) *User Stories Applied: For Agile Software Development* [online]. First Edition. Boston: Addison-Wesley Professional [Accessed 08/03/20].

Stellman, A (2019) *What is Scrumban?* [online] Academic Edition. Sebastopol. O’Reilly Media Inc. [Accessed 08/03/2019].

Pahuja, S (2015) What is Scrumban?. Available from: <https://www.agilealliance.org/what-is-scrumban/> [Accessed 10/03/2019].

Atlassian Agile Coach (2020) *Kanban vs. Scrum.* Available from: <https://www.atlassian.com/agile/kanban/kanban-vs-scrum> [Accessed 16 February 2020].

Klaus, L. (2015) *Kanban and change leadership: creating a culture of continuous improvement* [online]. New Jersey: John Wiley & Sons Inc. [Accessed 10 February 2020].

Emerson, C. (2020) Personal Kanban: A Tactile To-Do List to Make You a Project Management Pro. *Unemployable* [blog]. 12 January. Available from: <https://unemployable.com/personal-kanban/> [accessed 26 February 2020].

Lonergan, K (2019) *Why Scrum alone does not work, and, what will work much better.*  Project Management Informed Solutions [blog].

Beck et all (2001) *Manifesto for Agile Software Development* [online]. Available from: <https://agilemanifesto.org/> [Accessed 10 February 2020]. March 31. Available from: <https://www.pmis-consulting.com/why-scrum-alone-does-not-work/> [Accessed 07/03/2020].

Eby, K. (2016) *Comprehensive Guide to the Agile Manifesto* [online]. Available from: <https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto> [Accessed 10 February 2020].

Smartics (2015) *Agile Documentation* [online]. Available from: <https://www.smartics.eu/confluence/display/ADOC/Agile+Documentation> [Accessed 14 February 2020].

Linders, B (2014) *Documentation in Agile: How Much and When to Write It?* [online]. Available from: <https://www.infoq.com/news/2014/01/documentation-agile-how-much/> [Accessed 10 February 2020].

Selic, B. (2009) Agile Documentation, Anyone?. *IEEE Software* [online]*.* 26 (6), pp. 11-12. [Accessed 14 February 2020].

Moreira, M (2013) Right-sizing Documentation in an Agile World. *Agile Adoption Roadmap* [blog]. 15 June. Available from: <http://cmforagile.blogspot.com/2013/06/right-sizing-documentation-in-agile.html> [Accessed 10 February 2020].

Dinwiddie, G (2014) *The Three Amigos Strategy of Developing User Stories.* Available from: <https://www.agileconnection.com/article/three-amigos-strategy-developing-user-stories> [Accessed 10 February 2020].

Collins, E., Lucena, V. (2012) Software Test Automation practices in agile development environment: An industry experience report. *2012 7th International Workshop on Automation of Software Test (AST).* 2-3 June 2012. Zurich, Switzerland: IEEE Available from: <https://ieeexplore.ieee.org/document/6228991/footnotes#footnotes> [Accessed 16 February 2020].

Denys, I (2019) Epics vs User Stories: the Key Difference and Examples. *GBKSOFT Software Development & Consulting.* 21 June. Available from: <https://gbksoft.com/blog/difference-between-epics-vs-user-stories/> [Accessed 22/04/2019].

Dagka, F et al. (2010) Wikis in enterprise settings: a survey. *Enterprise Information Systems* [online]. 6 (1), pp. 1-53 Available from: <https://www.tandfonline.com/doi/citedby/10.1080/17517575.2011.580008?scroll=top&needAccess=true> [Accessed 20 February 2020].

Macleod, A (2020) Showcase, Show & Tell or Sprint Review. Scrum.org [blog]. 17 January. Available from <https://www.scrum.org/resources/blog/showcase-show-tell-or-sprint-review> [Accessed 20 February 2020].

Cagley, T (2013) Agile Testing: User Acceptance Testing. *Spam Cast* [online] Available from: <https://tcagley.wordpress.com/2013/08/02/agile-testing-user-acceptance-testing-august-2-2013/> [Accessed 14 February 2020].

Gupta, S (2016) Difference Between Pilot and Beta Testing. *Oodles Technologies* [blog]. Available from: <https://www.oodlestechnologies.com/blogs/Difference-Between-Pilot-and-Beta-Testing/> [Accessed 15 February 2020].

Ventayol, A (2016) Top Reasons Why Beta Testing Is Important. *Bugfender* [blog]. Available from: <https://bugfender.com/blog/top-reasons-beta-testing-important/> [Accessed 15 February 2020].

International Organisation for Standardisation (2017) *ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.* Geneva: International Organisation for Standardisation.

# 8.0 Appendix

**Appendix 1 –** **Proposal of application features from SOMO**

Your future finance portal dashboard (pre-authorised/logged in) with Key control Module, BU Manager view of the key control Dashboard (Perform/Review Controls), Data visualisation for the BU Manager role, View a list of your own KC’s, along with status, Type and Due Date, View a list of KC’s that have been flagged, View contracts, View KC details, Link to PCMB, View a checklist of tasks per KC, Select all tick boxes in KC checklist, Read all previous comments against a KC task, Add a new comment against a KC task, Add a flag against a KC task, Upload documentation/evidence to a KC, Submit a KC for approval, Receive a notification that a KC is ready to approve.