# The Sync-Up Process to Improve the Multiple Stakeholder Communication of Requirements Analysis in Embedded Medical Software Development

Surafel Demissie

Supervised by:

Dr Frank Keenan

Prof Fergal McCaffery

*A thesis submitted in fulfilment of the requirements for the degree of Master of Science to the*

DUNDALK INSTITUTE OF TECHNOLOGY

SCHOOL OF INFORMATICS AND CREATIVE ARTS

September 28, 2021

# Declaration

We, the undersigned declare that this thesis entitled The Sync-Up Process to Improve the Multiple Stakeholder Communication of Requirements Analysis in Embedded Medical Software Development is entirely the author's own work and has not been taken from the work of others, except as cited and acknowledged within the text.

The thesis has been prepared according to the regulations of Dundalk Institute of Technology and has not been submitted in whole or in part for an award in this or any other institution.

| | |
|---|---|
| Author: | Supervisor: |
| .......................................... | .................................. |
| *Signature* | *Signature* |
| .......................................... | .................................. |
| *Date* | *Date* |

# Acknowledgements

Firstly I would thank my beloved wife Fikirte who was always by my side and supporting me during the challenging time of my study. I would also like to thank my family specially my mother Meseret for her love and support throughout my life.

I would also like to thank my supervisory team, first and foremost, Dr Frank Keenan for his excellent supervision and continued patience with me throughout the duration of this project. Also, Prof Fergal McCaffery, who gave me the chance to be part of the RSRC research team and whose own work continued to inspire me.

I would like to thank the companies who participated in this research. A special thanks goes to Dermot Barron for his support at every stage of this research, particularly in the implementation of the Sync-Up process.

Finally, to all my colleagues and friends, thank you for your encouragement and support.

# Contents

# List of Figures

# List of Tables

# Publications

1. The Sync-Up Process to Assist Multiple Stakeholder Communication of Requirement Analysis in Embedded Medical Software Development – ICICT 2021. The paper presented the result of the case studies conducted to evaluate the proposed process.

2. Improving Multi-domain Stakeholder Communication of Embedded Safety-critical Development using Agile Practices: Expert Review. In Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development – MODELSWARD 2020. *This paper presented the result of the expert review conducted to evaluate the proposed process.*

3. Agile Usage in Embedded Software Development in Safety Critical Domain–A Systematic Review, International Conference on Software Process Improvement and Capability Determination - SPICE 2018. *This paper presented the result of the systematic review that was conducted to investigate the challenges related to agile usage in embedded safety critical domains.*

4. Supporting Embedded Medical Software Development with MDevSPICE® and Agile Practices, International Journal of Computer, Electrical, Automation, Control and Information Engineering - ICCSE 2017.

5. Investigating the Suitability of Using Agile for Medical Embedded Software Development, International Conference on Software Process Improvement and Capability Determination - SPICE 2016.

# Abstract

The development of embedded medical software is different from ordinary software development as it needs to be coordinated with the hardware development. A typical embedded system project involves multiple stakeholders such as the business unit, software developers, hardware engineers and firmware developers. Agile methods have been successfully adopted in generic software engineering, and more recently in embedded medical software development.

In this research, a systematic review has been performed to identify the challenges of embedded medical and safety-critical software development domains. From the challenges identified, this research focuses on the challenge of multiple stakeholder communication in embedded medical software development. Additionally, agile practices which have been successfully adopted in the embedded safety-critical domains have been investigated. This thesis describes the development and evaluation of a process (Sync-Up) to improve multiple stakeholder communication for embedded medical software development during requirement analysis.

Through this research, the following contribution to knowledge has been made in the area of embedded medical domain. The development of the Sync-Up process to assist multiple stakeholder communication of embedded medical software development. The Sync-Up process is evaluated through both expert review by leading experts, and a case study conducted in an embedded company. Findings from the evaluations undertaken show a positive outcome during the requirement analysis phase of the Sync-Up process.

# Chapter 1

# Introduction

## Introduction

An embedded system is a special purpose computer that is designed to perform a specific task with software stored on a chip that is referred to as firmware Kamal (2011). Today such systems are everywhere in our day-to-day life from household items such as a digital camera, refrigerator and TVs, to complex and critical devices like pacemakers and smart grid control units Vahid and Givargis (2000).

A typical embedded system consists of hardware and software. The hardware includes a microprocessor or microcontroller, memory, input-output (I/O) and additional components such as sensors and actuators which enable interaction with the environment. The embedded software is application-specific as it is dedicated to perform pre-designed specific tasks repeatedly, thereby controlling the functionality of the hardware device. Functions in such software are generally activated by external controls or remote input of data. Successful development here requires hardware devices to be available. Another characteristic is the necessity for real-time responses. Also, limited memory and power resources need to be considered.

Because the users of the embedded system do not need to intervene in the operation, except for the simple reset operation, once the embedded software is loaded into the system, the software is “expected to run for a very long time by itself without any changes to the software” Qian et al. (2009). It is expected that such software has been developed and integrated with the hardware correctly.

The complexity and growth rate of embedded software has been increasing over the past number of decades. As more functionality is added onto systems devices are becoming increasingly sophisticated and interconnected making embedded software development more challenging Zhang et al. (2014). A study by Teich (2012) states that today a single vehicle can contain more than 100 million lines of code. “The complexity of many modern embedded systems is at the level of distributed communicating electronic devices such as electronic control units (ECUs)”. In a single car, as many as 70-90 ECUs communicate with one another to provide special services such as stability control, antilock braking, or entertainment functions.

A key area in this sector is embedded medical software. Such software can be as simple as software running under a digital thermometer that has display, reset or switching operations or a complex software running under an implantable device such as a pacemaker or Magnetic Resonance Imaging (MRI). According to recent reports, the medical device market is expected to grow over the next five years with sales figures expected to expand from 470.5bn USD in 2018 to 640.9bn USD in 2023 BCC (2018).

Reflecting the critically of such devices, the embedded medical software development process is under the regulation of various international standards Hrgarek (2012). Based on their geographical location, medical device companies must follow the required international standards and guidance documents before marketing their products Munzner (2003). As such, they are obliged to conform to regulations outlined by Medical Devices Regulation (MDR) in Europe or Food and Drug Administration (FDA) in the US. In the EU for example, medical companies must have CE mark on their products to show that they have passed the conformity assessment (British Standards Institution-BSI). Audits will be performed by the regulatory body of the specific country.

Such an audit process includes satisfying the following standards: medical device quality management standard (EN ISO 13485:2016), medical device risk management (EN ISO 14971:2019) and medical device product level standard IEC 60601-1 (2005). Although beneficial, demonstrating conformance to such standards brings additional challenges to the embedded software development process.

In order to deal with the complex regulation process in an organised and structured manner, medical companies have been implementing traditional software development process models such as Waterfall Royce (1970) or V-Model Forsberg and Mooz (1991). Given the plan-driven nature of such models, with clearly defined milestones and associated artefacts, regulations and audits are easily facilitated at each checkpoint. However, such models are reported to be risky, and their implementation invites failure Royce (1970) Munassar and Govardhan (2010). For example, if requirements change during the development phase, the amount of rework is costly. Another key factor is the limited customer involvement which may result in miscommunication between development team and customers. Additionally, in these models, the implementation comes late in the process. This will make the expected results invisible for a long time with “no early prototypes” which can be “disconcerting to management and customers” Munassar and Govardhan (2010).

## 1.1 A Development Process for Embedded Systems

A software development process can be defined as the “description of phases in the product life-cycle through which the software is being produced” Abrahamsson et al. (2002). Generally, the phases of the software development process are requirements definition, architectural and detailed design, implementation, testing and maintenance Munassar and Govardhan (2010).

The development of embedded software depends on the corresponding hardware development process as it must interact with hardware components such as sensors and actuators. Such a parallel development process of hardware and software is known as Co-design Teich (2012) and Wolf (1994). Berger (2002) extended the traditional V-model to suggest a seven-phase embedded system design life-cycle. In this life-cycle, hardware and software engineers work together during the initial "specification" phases but then follow separate V-models. Hardware design path activities are followed on one side of the V-model and the software design path activities on the other side of the V-model. Such parallel development activities include diverse stakeholders such as hardware engineers and software engineers that must have effective communication and knowledge sharing. The two teams re-join to complete "integration" and "acceptance testing" phases.

Typically, Embedded Medical Software is produced through following a plan-driven process. The V-Model is commonly used as it involves decomposition of requirements and the creation of system specifications on the left-side and emphasises integration of parts and validation on the right Rottier and Rodrigues (2008). Having software and hardware development processes, the embedded medical domain has been implementing this model as "it appears to be the best fit with regulatory requirements" Mc Hugh et al. (2013) and enables system level integration and acceptance testing.

## 1.2 Agile in Embedded Medical & Safety-Critical

The development of embedded medical and safety critical software must deal with challenges at a high-level concerning certification and regulation and technical challenges associated with embedded system at a lower level. One approach that may offer assistance is Agile Methods (AMs) which has been a hot topic in embedded medical and safety critical domains in recent times.

AMs are an umbrella of software engineering methods that are based on iterative, incremental, and evolutionary software development process Greer and Hamon (2011). AMs have been in use in the application software development domain for the last two decades Dingsøyr et al. (2012). Generally, agile methods recommend a high degree of expert customer involvement, the ability to incorporate changing requirements and short development cycles producing working software. Numerous agile methods are available including eXtreme Programming (XP) Beck and Andres (2005), Scrum Schwaber and Beedle (2001), Feature Driven Development (FDD) Palmer and Felsing (2001), Dynamic Systems Development Model (DSDM) Stapleton (2014), Lean Mary Poppendieck (2003) and DevOps Erich et al. (2014). Although each employs different practices, all adhere to the Agile Manifesto Beck et al. (2001).

Previous studies of agile implementation in embedded medical and safety critical domains report both benefits and challenges Xie et al. (2012). This is because the development of embedded software is different from commercial or application software as it has to interact with hardware in real-time Woodward and Mosterman (2007). While commercial software development focuses on algorithm and data processing, embedded software development aims at managing and controlling the system or hardware.

Existing literature covers agile usage and challenges in the safety critical domain and embedded system themes separately. In this research, the challenges of agile usage in the embedded medical domain will be investigated. The research is formulated based on research questions and objectives outlined in the following section.

## 1.3 Research Questions and Objectives

This research is formulated based on the Initial Research Questions (IRQ) and the Refined Research Question (RRQ). To answer these research questions, a number of research objectives were formulated. The IRQ of this research are defined as follow:

IRQ.1 : What are the challenges related to agile implementation in embedded safety-critical software development?

IRQ.2 : What agile practices have been used, and how are the practices implemented in embedded safety-critical software development?

Based on the findings of the initial research questions, a specific focus was applied to address one of the challenges identified for embedded medical software development. The following Refined Research Question (RRQ) was formulated.

RRQ : How can we support multiple stakeholders communication during requirements analysis of embedded medical software development using a combination of suitable agile practices?

To address the IRQ and RRQ, the following research objectives were formulated:

OBJ.1 : To identify the challenges affecting an agile implementation in embedded safety critical domains and investigate the improvement recommendations.

OBJ.2 : To identify suitable agile practices and investigate their implementations in embedded medical and other embedded safety critical domains.

OBJ.3 : To develop a process using a combination of suitable agile practices, that assist the multiple stakeholder communication of embedded medical software development.

## Aim

The aim of this research is to support the challenge of multiple stakeholders communication of embedded medical software development using a combination of suitable agile practices. From the systematic review (SR), this research identified suitable agile practices that have been preferred in embedded medical and other embedded safety critical domains. In order to support multiple stakeholder communication, the Sync-Up process has been developed using a combination of suitable agile practices.

## The Sync-Up Process

The Sync-Up process has been developed to assist multiple stakeholder communication during requirements analysis in embedded medical software development. The process is developed by combining agile practices identified as most suitable through the SR. The overall approach is based on the foundation of Acceptance-Test Driven Development (ATDD).

The Sync-Up process has been reviewed by leading experts in the embedded safety-critical domain, and suggestions and recommendations were addressed to evolve the process. The validation of the process was conducted through exploratory and confirmatory case studies. The exploratory case study was conducted in an academic setting while the confirmatory case study was conducted in an embedded company.

The overall approach of the research including the research questions and objectives is illustrated in Figure 1.1.

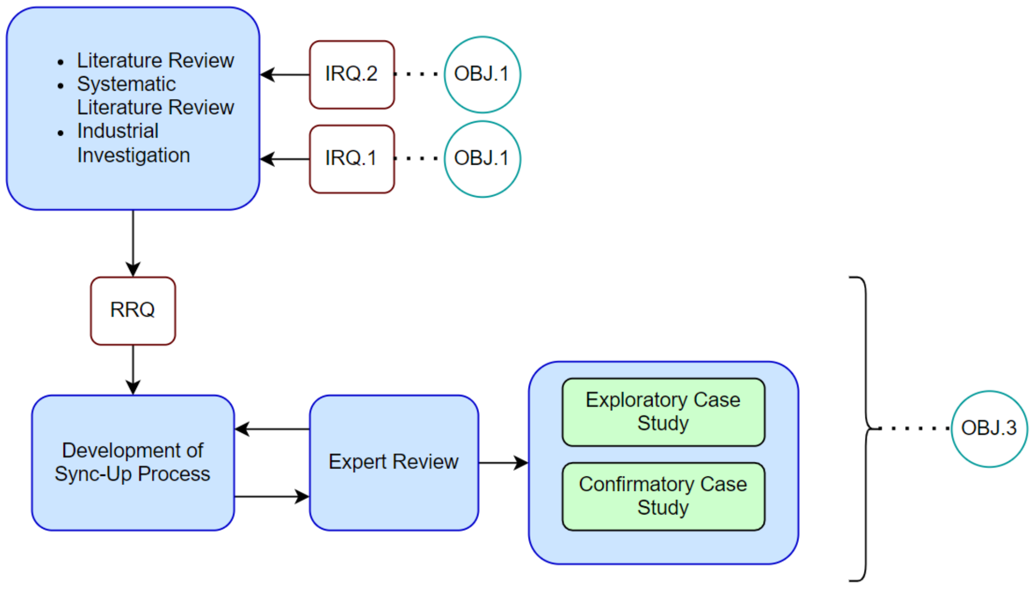


**Figure 1.1:** Research Approach Overview

## 1.4 Document Outline

The document contains seven chapters. Chapter 1 is this introductory chapter. Chapter 2 presents the research background that consists of the systematic review and industrial investigation that was performed with embedded companies. This is followed by Chapter 3, which is the research setting. Chapter 4 presents the description of the Sync-Up process. Chapter 5 presents the expert review conducted with leading experts in embedded and agile software development domains. Chapter 6 presents the implementation that covers the case studies conducted to evaluate parts of the proposed process. Finally, the summary and conclusion will be presented in Chapter 7.

# Chapter 2

# Research Background

## Introduction

This chapter presents the literature review and industrial investigation conducted to investigate the challenges of embedded medical software development. The chapter begins with an overview and development process of embedded systems. Thereafter the challenges of embedded medical software development will be presented. Subsequently, the review of agile methods and practices will be presented. Finally, the systematic review, which is conducted with a particular focus on the challenges related to agile usage in embedded medical and other embedded safety-critical software development domains, will be presented.

## 2.1 Embedded Systems

According to Marwedel (2006), "embedded systems are information processing systems embedded into enclosing products such as cars, telecommunication or fabrication equipment". The system is composed of software and hardware components to carry out a specific function. The software inside an embedded system usually embeds into flash memory or read-only memory (ROM). Most embedded systems are a subset of a larger system. For example, modern cars and trucks have many embedded systems to control the anti-lock brakes, the vehicle's emissions, and to display information on the dashboard Barr (1999).

Embedded system applications have become an integral part of human lives. Applications such as home security and alarm systems, thermostat, washing machines, cars, traffic light, elevators, printers, digital watches, digital cameras and automatic teller machines (ATMs) are some examples of embedded systems Barr (1999); Bolton (2000); Fisher et al. (2005).

Embedded systems interact with the external environment using sensors and actuators to control or respond to certain behaviours. As a result of the interaction with the a real-world, embedded systems are also called real-time systems. In real-time systems, the correctness of the system depends "not only on the logical results of the computations but also on the physical time when these results are produced" Kopetz (2011). Interaction involves receiving data or input command, processing data, and responding to the environment within a reasonable time-frame. Based on the response time, real-time embedded systems are categorised as *hard* and *soft* real-time embedded systems. Hard real-time embedded systems must perform all computation and respond to the environment in *strict time*. Such systems must also "sustain a guaranteed temporal behaviour under all specified load and fault conditions" Kopetz (2011). Hard real-time embedded systems are mostly used in safety-critical applications. Some examples of hard-real time systems include anti-lock braking system, cardiac pacemaker, and Antimissile system. Soft real-time embedded systems, on the other hand, have a *less strict* response time. For these systems missing a deadline is usually considered tolerable. Some examples of soft-real time systems include digital cameras, wireless router, and global positioning system.

The development of embedded systems "requires a holistic approach that integrates essential paradigms from hardware design, software design, and control theory in a consistent manner" Henzinger and Sifakis (2006).

### 2.1.1 Components of Embedded Systems

An embedded system in its simplest form consists of memory, input-output (IO) interfaces, application-specific processor and software to invoke the required operation. Figure 2.1 shows the structure of a generic embedded system as defined by Barr (1999). A short summary of each component will be explained in the following subsections.

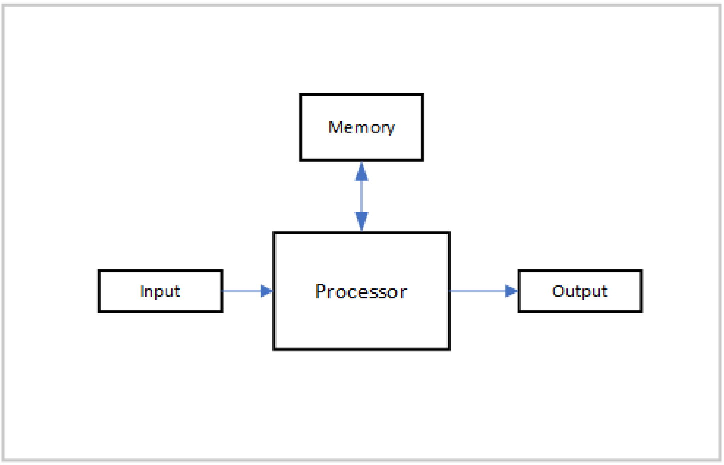


**Figure 2.1:** Generic Embedded System defined by Barr (1999)

#### Memory

Memory is a hardware component that stores data temporarily or permanently. Memories that store data temporarily are also called volatile memories, and the ones that store permanently are called non-volatile memories. The embedded system needs the volatile memory to store program variables and intermediate results and to manage data structures such as the stack. Data stored in volatile memories will be lost if the power of the system is turned off. The non-volatile memories are used to store data permanently. These memory types are usually used to store permanent data that needs to be retrieved after power is switched off. Applications such as an operating system are stored in these memories.

### IO Interfaces

IO interfaces are hardware components that connect the embedded system with the outside world. Input interfaces can be sensors and probes, communication signals, or control knobs and buttons. On the other hand, output interfaces typically display communication signals or changes to the physical world. Output interfaces have different variations such as binary output, serial output, displays and time derived outputs.

### Processor

A processor is a basic unit that takes inputs and produces an output after processing the data or computing. There are two main types of processors. These are microprocessor and microcontroller. A microprocessor is a programmable chip that has the capabilities of computing and decision making. They have the arithmetic and logic unit (ALU) and a control unit (CU) to process the execution of instructions. A microcontroller is a programmable digital processor with necessary peripherals and on-chip memory. They can be thought of as a microprocessor with additional peripherals and on-chip memory.

### Embedded Software

Embedded software is an entity that is tied to a specific hardware device with the role of interacting with the physical world. Unlike application software, embedded software has less visibility and fixed hardware requirements. The developers of embedded software are experts in the application domain with a good understanding of the target hardware that includes reading schematics and datasheets. A specific class of embedded software that helps to control and access the hardware is called embedded firmware. They are called firmware as a result of the difficulty in upgrading or fixing bugs. Appliances such as microwave ovens, refrigerators and televisions have embedded firmware.

### 2.1.2 Roles In Embedded System Development

A typical embedded system project can contain a business team and technical experts. The business team is composed of *Customer* representative. The technical experts can be composed of *Hardware Engineers, Firmware Engineers, Software Developers* and *Testing professionals.* Figure 2.2 shows the overall roles that are involved in an ideal embedded system project. On the next subsections, a short summary of each role will be presented.

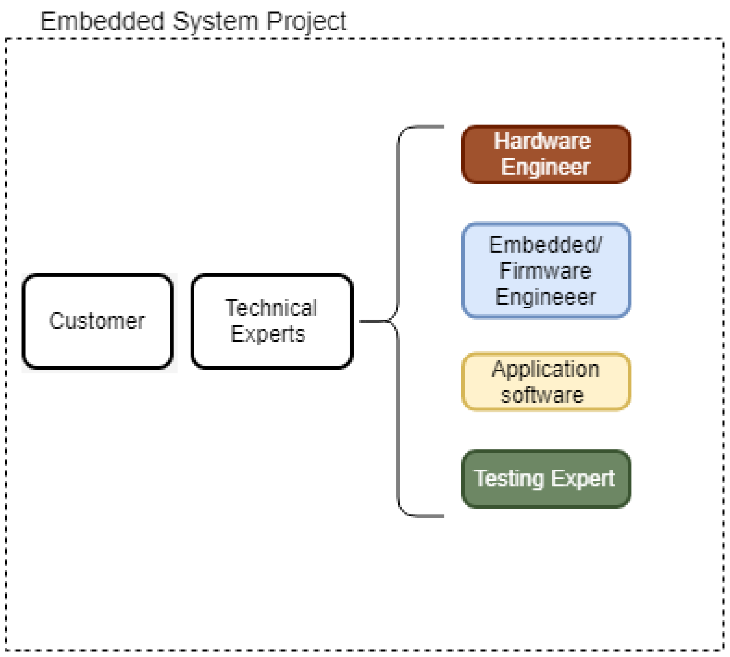


**Figure 2.2:** Roles in Embedded Software Development

#### Customer

A customer represents a person that has a business understanding of the product to be developed. The customer will be involved during the duration of the project to provide guidance on project requirements, User stories and acceptance testing.

#### Technical Experts

The technical experts in embedded system includes hardware engineer, embedded/firmware engineer, application Software engineer and testing expert.

**Hardware Engineer**

Hardware engineers design and develop the functions of the dedicated computer chips and systems. The tasks of the hardware engineer include studying the data-sheets of various components and peripherals from the point of view of their electrical characteristics, designing the schematic of the printed circuit board (PCB) for the product, designing the power section of the device, board bring-up and testing when the PCB arrives and solving any hardware related issues with other developers.

**Embedded (Firmware) Engineer**

The embedded/firmware engineer design and develop an optimised code for specific hardware platforms. The tasks of the embedded engineer include writing very well-versed device drivers and microprocessor interfacing. Depending on the application, the engineer is also expected to understand the system on chip (SOC) and different peripheral devices.

**Application Software Engineer**

Software developers are responsible for the development of the application software that interacts with the embedded device. The software development engineer can be involved from requirement elicitation to development and acceptance testing. The application software development for an embedded system can include database development and User Interface (UI) design.

**Testing Expert**

The testing expert conducts the overall functional testing of the product. Additionally, the testing expert will set up test environments and conduct automated and manual tests by designing testing scenarios.

## 2.2 Embedded System Design Process and Challenges

### 2.2.1 Embedded System Design Process

The design of embedded systems requires a holistic approach that integrates essential paradigms from hardware design and software design. A typical design process of embedded systems follows a "top-down design process" known as *hardware-software co-design*. In this approach, hardware and software components are designed concurrently by taking into consideration the cost, energy, performance and speed Wolf (1994) and Teich (2012). Figure 2.3 shows the "embedded system design cycle" defined by Berger (2002). The summary of the major phases will be presented in the next subsections.

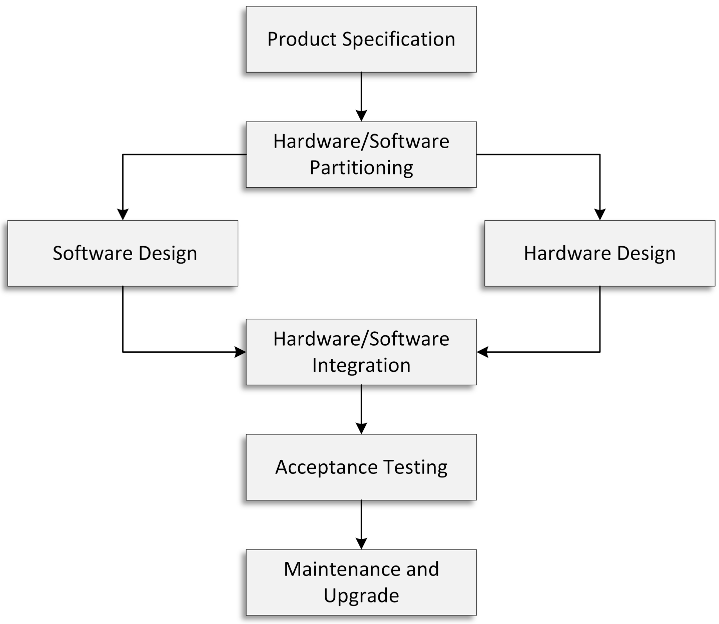


**Figure 2.3:** Embedded System Design Life cycle Berger (2002), Ernst (1998)

#### Product Specification

The first phase of the cycle involves defining a product specification composed of functional and non-functional requirements. The functional requirement includes the operations to be performed by the system. Non-functional requirements include speed, power, and manufacturing cost Wolf (1994). According to Berger (2002), the product specification has to involve the meeting between a customer, marketing or sales engineer and two or three R&D types through "customer visit tour" to turn a concept into a set of product requirements Berger (2002).

### Hardware/Software Partitioning

After defining the product specification, the next phase includes deciding the components/modules of the problem that will be solved in hardware and software Berger (2002). For an algorithm to be implemented, the "partitioning decision" will enable us to decide if the algorithm can be implemented purely in software, hardware, or in some combination of the two Berger (2002).

### Software Design

The next phase involves the separation of software and hardware design phases. The software design phase involves developing and running code benchmarks on single-board computers with specific microprocessor. The single-board computers are known as evaluation boards as they help to "evaluate the performance of the microprocessor by running test code on it". Having separate paths, the "hardware and software designers and system architects must synchronise their work progress to optimise and debug a system in a joint effort" Ernst (1998). The separation of the design phases is reported to create a major problem in synchronisation and integration of hardware and software Ernst (1998), Berger (2002).

### Hardware Design

The hardware design activities involve design engineers reading design schematic, designing circuits for functionalities, and using simulation tools to model the performance of the processor and memory systems. The next activity involves building schematics and layout a printed circuit board (PCB).

#### Hardware/Software Integration

The process of integrating embedded software and hardware is an exercise in debugging and discovery. The integration involves "combining the first hardware prototype, the application software, the driver code, and the operating system software together with a pinch of optimism and to have the design work perfectly out of the chute" Berger (2002).

#### Acceptance Testing & Maintenance

Acceptance testing involves the testing of the embedded product with respect to predefined requirement parameters such as performance. Acceptance testing of the embedded system is much "more stringent than the vast majority of desktop applications" Berger (2002). Maintenance involves upgrading existing products, rather than designing new products. The upgrade of existing products requires optimisation to create both hardware and software that complement each other, and this requires tools that are tailored to reverse engineering Berger (2002).

### 2.2.2 Challenges for Embedded System Development

The development of embedded systems brings challenges from embedded software development which is related to hardware dependency, real-time and resource constraints. Woodward and Mosterman (2007) identified five challenges that are believed to break the embedded software project trade-offs. These are complexity, optimisation, interdependency, verification, and tools. Complexity refers to the addition of more and more functionality onto a single system. Optimisation refers to the selection of the "best" combination of hardware and software elements that interact with each other. The challenge of interdependency comes from having different parts of the design process that are increasingly interdependent but keeping the design processes and design domains separate. The report states that a better way of "synchronising different design version must be developed." Verification refers to checking if the system implemented meets the specification. The study calls for better techniques of verification automation using system specification.

A systematic literature study by Rong et al. (2014) categorised factors affecting embedded system design in four categories. These are development factors, human factors, external factors and internal factors. The development factor includes a lack of proper methodologies. The study states that hardware-software partitioning is one of the challenging development factors that play an important role in system design. The human factors represent the skill developers should have in the diversified development environment. The report states that "close interaction among hardware and software developers was necessary yet unsatisfied for embedded systems design." The external factors are market pressure that affects time-to-delivery, which can introduce other experts from, for example, marketing and business. The category of internal factors includes hardware dependency, resource constraints such as memory, power consumption and response time.

A study by Graaf et al. (2003) reported the result of 36 interviews with software practitioners from seven industrial companies building embedded software products and one research institute. The study reports that most embedded software development are sub-processes of systems engineering, and coordinating such sub-processes is one of the most challenging aspects of developing embedded software. Additionally, systems engineering is mostly driven by hardware development because of longer lead times.

Another study by Ebert and Jones (2009) states that the "embedded software systems pose extraordinary challenges to the software engineer due to their complexity". The complexity comes from having a large number of interactions among the various parts of these systems. The report also states that "malfunctions of embedded software are much higher than those of application software" and embedded software engineers must use more extensive defect prevention activities than other software domains.

### Summary of Challenges

The summary of challenges associated with the development of an embedded system are:

- Interaction - refers to the interaction that software, embedded and hardware engineers need to have to deliver the embedded system. It was reported that close interaction is required between these diverse members and studies calls for better ways to assist collaboration.
- Interdependency - comes from having different parts of the design process that are increasingly interdependent.
- Lack of proper methodologies - it was reported that the embedded system design is affected by the lack of proper methodologies that combine hardware/software.
- Hardware dependency - the embedded system design is affected by the close ties of to hardware development.

In order to understand the embedded medical software development process and explore the challenges, an industrial investigation was conducted with two embedded medical companies.

## 2.3 Industrial Investigation

The industrial investigation was conducted with two companies that develop embedded medical software. The first company, Company A, produces high-performance aerosol drug delivery technology for hospitals for better patient care. The second company,

Company B, produces a global positioning system (GPS) player tracking and analysis equipment.

In order to conduct the industrial investigation, semi-structured interview questions were developed. Semi-structured interviews are composed of open-ended and specific questions that are "designed to elicit not only the information foreseen but also unexpected types of information" Shull et al. (2007). The interview questions were developed around the process of developing embedded medical software following the guidance of (Dawson. C, 2008, pp.69) and Brace (2018). The interview questions have been reviewed by supervisors, and comments have been addressed on irrelevant topics. The list of questions is shown in Appendix A.

### 2.3.1 Interview with Company A

The interview was conducted with two engineers. The first engineer is a senior electronic engineer while the second one has software engineering experience. As both interviewees have declined to record the interview, detailed hand-written notes were taken and immediately transcribed following the interview. In the following subsections, the responses of the interviewees will be summarised.

#### Development Team Structure

At the time of the interview, the development team was composed of design assurance (a dedicated person responsible), one lead software developer, a second software engineer, that acts as a reviewer of the development steps, and a technical design engineer that executes the validation and verification test scripts.

### Requirement Definition and Architectural Design

The interviewees stated that initially, the marketing team collaborate with stakeholders to develop the design input and, marketing requirements document (MRD). The stakeholders are composed of number of members from lead engineer, development team and hazard analysis. The MRD will then be distilled into a software requirement document (SRD). The SRD defines the simple basic description of the device functionality such as monitoring of voltages and currents.

At the time of defining MRD, if the development team realised that they couldn't implement a particular functionality, that was defined by the marketing team, they would have to inform the marketing team to update the MRD. One of the interviewees also stated that changes in the hardware would also have consequences on the MRD. For example, a timer that was set for 4 hours would have required 6 hours when they are implementing different algorithms.

Moving forward, the interviewees suggested that they can see problems relating to the user interface when they are developing firmware. The user interface is anything that the user interacts with such as display, touch screen, buttons and knobs, cover, cap, and cases. They suggested that addressing different stakeholder demands on the user interface and making everyone happy would be challenging.

### Implementation, Testing and Maintenance

Once they have all controls defined on the SRD, the team will develop software and hardware in parallel with the printed circuit board (PCB) layers and first fabrication prototyping. The implementation is based on phase-based modules which include developing the control parts of the circuit, control of every other function layers of the hardware, developing a phased version of the code and implementing functional and test coverage. The interviewees stated that before software developers start coding, the hardware with the desired specifications of timing and frequency should be available. If the hardware is not available, the software development activities need to wait for the hardware to be delivered for integration and testing to be performed at a low level.

When asked if the company have used generic off the shelf evaluation boards, the interviewees stated that if they have concerns on specific modules, sometimes they use similar hardware that was developed in-house from previous projects and they then perform tests on this alternative hardware while waiting for the actual hardware to be delivered.

According to the interviewees, during the early phases of the development, sometimes hardware-related bugs affect software development. Bugs such as the power supply not acting as it should and noise on the board affecting measurement and resulting in the incorrect functionality of the algorithm.

In addition to hardware development, the development of embedded medical software is also affected by the availability of mechanical parts. One of the interviewees stated that for example, a breath detection system needs the mechanical parts to run the software testing and get the parameters correct for the algorithm. Subsequently, this could cause some delays to the overall development process. The interviewee also highlighted that the weekly meetings are used to track such delays.

#### Risks Analysis

Hazard and risks analysis were part of the SRD. The interviewees stated that at the time of the interview, the company was considering splitting them from the SRD. From a product development perspective, the company has a system hazard user analysis document that identifies risks that could involve hospital visits and observation of the user. There was also another document, failure mode effects analysis (FMEA), that defines hardware failures that would have a consequence on software.

**Summary of Interview with Company A**

The key issues that have been identified from the interview were:

- Hardware, software, and mechanical development dependency - the interviewees stated that the embedded software development would be affected if hardware and mechanical parts were not delivered in time. Sometimes they use similar hardware in-house to test some modules while the actual hardware is developed.
- Multiple stakeholder demands - they have suggested that addressing the demands different stakeholders have on the user interface must include the overall hardware and embedded software functionalities and making everyone happy would be challenging.
- Managing changes in the MRD - they have suggested that the MRD will need to change if the development team could not perform the functionally in software. Changes in the hardware were also suggested to affect the MRD.

### 2.3.2 Interview with Company B

Company B, established in 2017, is based in Newry and develops performance tracking and real-time data analysis devices for elite sport clubs. The company produces products that are composed of hardware, firmware, and software components. The software component is used to extract data from the sensors. The interview was conducted with a Software Architect (former principal Software Engineer). At the time of the interview, the interviewee was working between two products which are team series and pro-series or elite version. Recording of the interview was allowed, and the recording was transcribed immediately following the interview. In the following subsections, the responses of the interviewee are summarised.

### Development Team Structure

At the time of the interview, the company had separate development teams for the consumer side and elite side products. They also had another developer in the team working on Android development. Additionally, the embedded firmware development was mostly looked after by an embedded engineer that was working remotely from Romania. In addition to the software developers and an embedded engineer, hardware engineers were also responsible for the development of hardware components such as APEX pods, docking stations and antennas that were produced in-house.

### Requirement Definition and Architectural Design

According to the interviewee, the initial phase of development is driven by the advances in the hardware. From the very start, software developers, the embedded engineer, the product owner, and product managers all come together during the initial phase and come up with the initial requirements, design and then refine from there. The interviewee highlighted that; user stories are defined only from the application software side. On the other hand, acceptance tests are conducted by sport scientists at the end, when the application software becomes ready.

### Implementation, Testing and Maintenance

Regarding implementation, the interviewee stated that hardware is the first component that will be developed. That will make its way to the embedded firmware development. The embedded engineer will collaborate with hardware designers and develop the firmware. Once the firmware is produced at a beta stage, the embedded engineer will forward the full specification of the protocol to the software developers. For example, for the athlete series product, the software developers will use the specification protocol to interface with the device through a Bluetooth low energy (BLE) connection and start developing the application software. Once the hardware and application software are ready, then the quality assurance (QA) tests and acceptance testing will be conducted.

The interviewee recalled one occasion on the team series product where narrowing down bugs, which might be either from software, embedded or hardware side, was challenging. On this product, data was collected and transferred through USB that plugs into a docking station and with twelve units going into it. The unit was throwing some configuration error. The application software development team looked at the error and then passed it down to the embedded developer, and then at some point, the embedded developer passed it to the hardware team. The issue took quite a number of months to solve. The embedded and hardware developers thought it was an issue with the application software, the software developers, on the other hand, thought it was a hardware issue. It turns out to be a bug in the firmware of the operating system that was running on the unit. The data type was getting bigger than it was supposed to be and roll over into a bigger data type. This bug was crashing the application software after that number was got above the threshold. The interviewee suggested that, if the user stories and acceptance tests were developed with the engagement of the embedded and application software engineers, they might narrow down the solution quicker.

Another challenge that was experienced by the interviewee was the remoteness of the embedded engineer. The interviewee stated that it would have been easier to discuss and resolve issues when the software teams and an embedded engineer can communicate face to face. But discussing over Skype, it was difficult to narrow down problems during the integration of the embedded firmware and the application software components.

**Summary of Interview with Company B**

The key issues that have been identified from the interview were:

- Dependency on hardware development - it was stated that the initial stage of the software development needs to wait for the specification protocol to be available.

- Bug tracking - tracking and narrowing down issues from the application and hardware side were difficult to manage.
- The remoteness of the embedded engineer - this has been reported to be difficult in general requirement discussions and when problems arise.

### 2.3.3 Summary of Industrial Investigation

The industrial investigation was conducted to investigate the process and challenges of developing embedded medical software. The interview conducted with both embedded companies revealed that the development of embedded medical software involves diverse technical stakeholders such as software developers, design engineers, embedded firmware engineers, hardware developers and QA engineers. The interviewees stated that the availability of such stakeholders created a challenge on putting multiple stakeholder demands of hardware and software into the software requirements. In one of the companies, where the embedded engineer was working remotely, general requirement discussions were reported to be challenging.

Additionally, interviewees from both companies also revealed that embedded medical software development is dependent on the availability of hardware and mechanical parts. The interview also revealed that the embedded medical software development has a different process than generic software development process as it must interact with the corresponding hardware and mechanical components. The availability of such additional development processes of hardware and mechanical components brings additional challenges when dealing with bugs that can be from the software side or hardware side.

## 2.4 Agile Software Development

### Introduction

This section will examine agile software development. Initially, the values and principles of agile software development will be discussed. Then a description of agile methods that were reported to be widely used will be presented.

The software development process involves a set of activities which leads to the production of a software product Somerville and Somerville (2015). Royce et al. (2009) categorised the trends in software engineering into three generations. These are waterfall (1960s - 1970s), process improvement (1980s - 1990s) and agile development (2000s and on). Agile software development has been the latest trend from the early 2000s.

The term *agile* method was coined in 2001 by a group of industry leaders and advocates of lightweight frameworks. All agile methods share four common values and twelve principles that have been defined in the agile manifesto Beck et al. (2001). Any method or approach that claims to be agile needs to consider the values and adhere to the principles.

### 2.4.1 Agile Values and Principles

The four values which are part of the agile manifesto advocates:

**Individuals and Interactions** over **Processes and Tools**

**Working Software** over **Comprehensive Documentation.**

**Customer Collaboration** Over **Contract Negotiation.**

**Responding to Change** Over **Following a Plan.**

The manifesto, Beck et al. (2001), states that "while there is value in the items on the right, agile manifesto value the items on the left more." For example, the first value of the manifesto states that the success of the team depends on the way individuals work together, their ability to communicate effectively and efficiently. Such interaction is more valuable than processes and tools. The second value, which highlights the importance of working software over documentation, prioritise getting the software to the customers than letting documentation be a bottleneck. This means documentation is also essential, but it is suggested not to overdo it. The customer collaboration over contract negotiation encourages building a continuous customer feedback loop over contracts which dictate what was delivered in the end. The fourth value prioritises responding to change through frequent reviewing of the current plan and having a dynamic strategy to address new information that the team gather.

### 2.4.2 Agile Methods and Practices

A methodology is a system of practices, techniques, procedures, and rules used by those who work in a given discipline ISO 24765 (2010). Practices are concrete activities and work products that a method defines to be used in the process. Abrahamsson et al. (2017) defined a process as a "description of phases in the product life-cycle through which the software is being produced".

AMs are an umbrella of software engineering methods that are based on iterative, incremental and evolutionary software development process. These methods have evolved to reflect industry feedback. For example, XP started off with 12 practices Beck (1999) but an updated version, XP2 Beck and Andres (2005), replaces these practices with 24 practices that are categorised as either primary or corollary. With the original version, XP1, the intention was that each practice was mandatory for each project. However, this has evolved, and XP2 has been modified to follow a phased adoption of XP practices.

All AMs encompass on the four core values of the agile manifesto. The latest report from VersionOne.Inc (2020), the largest and longest-running survey on agile, states that

Scrum and Scrum/XP hybrids constitute about 70 % of reported agile usage. Scrum on its own increased its prominence as the most popular agile method from 40% in the first survey in 2007 to 58% and 70% when combined with other methods in 2020. At the same time, Extreme Programming (XP) lost ground from being the second most popular method (23%) to being used in combination with Scrum at 6%. The report also states that 84% of organisations are "still maturing" in their agile practice adaption and calls for opportunities for improvement.

Kniberg (2015) on his book *Scrum and XP from the Trenches* stated that Scrum and XP can be combined because “Scrum focuses on management and organisation practices while XP focuses mostly on actual programming practices”. The author also stated that some XP practices that are directly addressed by Scrum can be seen as *overlapping*, and we can simply be stuck to Scrum.

Solinski and Petersen (2016) stated that Scrum and XP are the most popular and adopted methodologies. Additionally, combinations of the classic Waterfall/XP, and Scrum/XP have also been reported to be commonly used. Another study by Theocharis et al. (2015) reported that “agile and traditional approaches are used in a mixed-method approach”. Specifically, Scrum has been reported to be combined with the classic Waterfall model and V-shaped processes. In the following subsections, a summary of the two most common agile methods and their respective practices will be presented.

### 2.4.3 eXtreme Programming (XP)

XP was formulated in the late 1990s when Kent Beck was working on the Chrysler C3 payroll project. Beck defined XP as “a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements” Beck (1999). The term *extreme* is used to reflect the concept of taking software engineering practices to extreme levels Beck and Andres (2005). The methodology emphasises teamwork and customer involvement throughout the software development process. XP provides a faster time to market, higher quality software, better customer satisfaction, and highly motivated development teams. It is developed based on a set of values, principles, and practices that helps developers to improve the software quality. XP has five core values which are communication, simplicity, feedback, courage & respect. On the other hand, the principles of XP serve as a bridge between the values and the practices. These principles are rapid feedback, assumed simplicity, incremental changes, embracing change and quality work.

### Roles in XP

According to Beck (1999), XP has the following major roles:

- ***Customer*** - the customer is responsible for writing user stories and sets priorities on implementation. He picks for them for product releases and iterations.
- ***Programmers*** - the programmers write the code, tests, refactors code, identifies tasks to be performed and estimates the time to perform them.
- ***Testers*** - this team member is responsible for helping the customer to write acceptance tests.
- ***The Coach*** - the coach checks the evolution of the process and watches the team's work in following the process.
- ***Doomsayer*** - this member tracks the project risks and warns the team about them.

### The XP life cycle

The life-cycle of XP is composed of six major phases. As shown in Figure 2.4 the phases are exploration, planning, iteration to release, productionizing, maintenance and death.

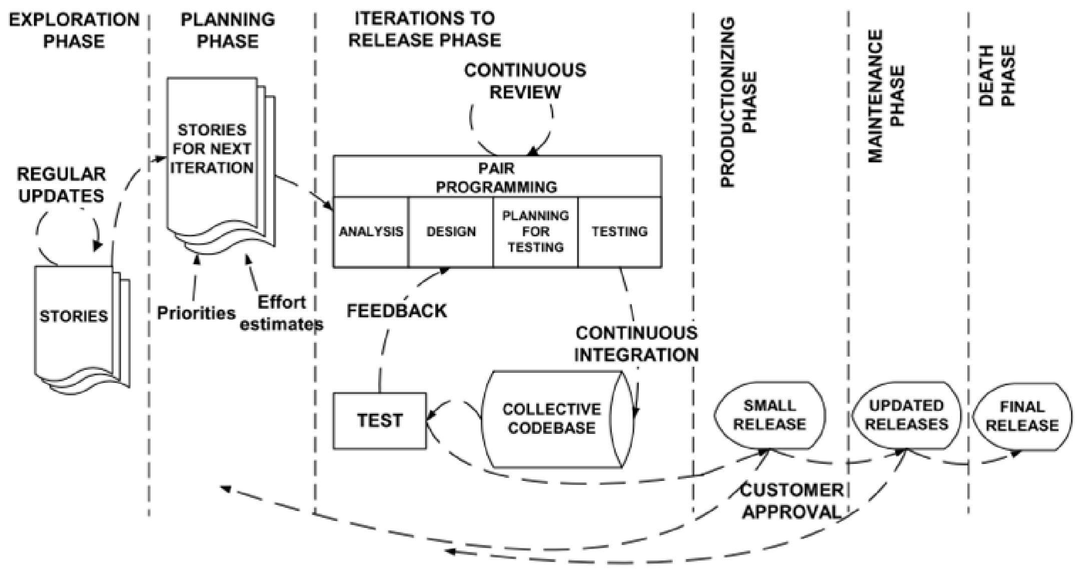


**Figure 2.4:** The Life cycle of the XP process (Abrahamsson et al. (2017))

- ***Exploration Phase*** - in this phase the project team will conduct exploration activities such as familiarising themselves with the tools and technologies they will be using and exploring architectural possibilities and prototypes for the system. The customer on their side will write story cards that will be part of the first release.

- ***Planning*** - in this phase, programmers estimate each story, prioritise and select the stories that will be involved in the first small release. The customer will be contacted for all aspects of the software feature. During this phase, the team will set goals for the entire project.

- ***Iterations to Release*** - this phase involves the implementation of a number of iterations before the first release. This involves the creation of the whole system architecture. This phase will generate a system that is ready for production.

- ***Productionizing*** - in this phase performance and extra testing activities will be conducted using the acceptance test. The team will work on stabilising the product and tuning performance.

- ***Maintenance*** - after the production phase, the maintenance phase will run to work on enhancements and bug fixing.

- ***Death*** - in this phase, the customer will confirm the correctness of the system developed, and there are no stories left. Documentation of the system will also be written. The death phase may also occur "if the system is not delivering the desired outcomes, or if it becomes too expensive for further development" Abrahamsson et al. (2017).

### Practices of XP

The first version of XP defined 12 practices Beck (1999). In the second version of XP Beck and Andres (2005) defend 24 practices categorised as primary and corollary practices. Beck suggested that the primary practices must be applied first. It will be difficult or dangerous to implement corollary practices before completing the preliminary practices. Table 2.1 summarises the first and second version of the practices of XP.

| **XP1** | **XP2** | |
|---|---|---|
| | **Primary Practices** | **Corollary Practices** |
| The Planning Game | Sit Together | Daily Deployment |
| Small Releases | Whole Team | Single Code Base |
| Metaphors | Informative Work-space | Shared Code |
| Simple Design | Energised Work | Code and Tests |
| Testing | Test-First Programming | Root-Cause Analysis |
| Refactoring | Incremental Design | Shrinking Teams |
| Pair Programming | Pair Programming | Team Continuity |
| Collective Ownership | Quarterly Cycle | Pay-Per-Use |
| Continuous Integration | Continuous Integration | Real Customer Involvement |
| Sustainable Pace (40-Hour Week) | Slack | Incremental Deployment |
| Whole Team | Quarterly Cycle | Negotiated Scope Contract |
| On-Site Customer | Stories | |
| Coding Standards | Weekly Cycle | |
| | Ten-Minute Build | |

**Table 2.1:** First and Second Version XP Practices

## 2.4.4 Scrum

Scrum is a process framework that helps teams "to manage work on complex products" and address complex adaptive problems. The framework was initially developed by

Jeff Sutherland in 1993 for managing and developing software products combining concepts from a previous article, *the new product development game* Takeuchi and Nonaka (1986), object-oriented development, empirical process control, iterative and incremental development and complex adaptive systems Schwaber and Sutherland (2017), Rubin (2012). The Scrum framework is composed of roles, activities and artefacts (Practices).

- Roles - the roles of the scrum framework includes *Product Owner (PO)*, *Scrum Master (SM)* and *Development Team*.
  - PO - the product owner is a person that is "responsible for managing product backlog". Additionally, the PO is also responsible to "ensure that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next" Schwaber and Sutherland (2017).
  - SM - the SM is a facilitator that maintain the Scrum processes to maximise the value of the scrum team and ensure that "the Scrum Team adheres to Scrum theory" Schwaber and Sutherland (2017).
  - Development Team - the development team includes the rest of the team that performs the actual software development.
- Activities and Artefacts (Practices) - the scrum activities includes sprint planning meeting, product backlog grooming, daily Scrum, sprint review and sprint retrospective. On the other hand, scrum artefacts are product backlog and sprint backlog.

Figure 2.5 shows the Scrum framework, adapted from Sutherland and Schwaber (2014), with roles, activities and artefacts. The framework starts with a vision of a product owner that will be broken down into a set of features called product backlog. The product backlog items will then be refined, estimated, and prioritised using the activity product backlog grooming. Rubin (2012).

Once backlog items are prioritised, the sprint starts with a sprint planning meeting

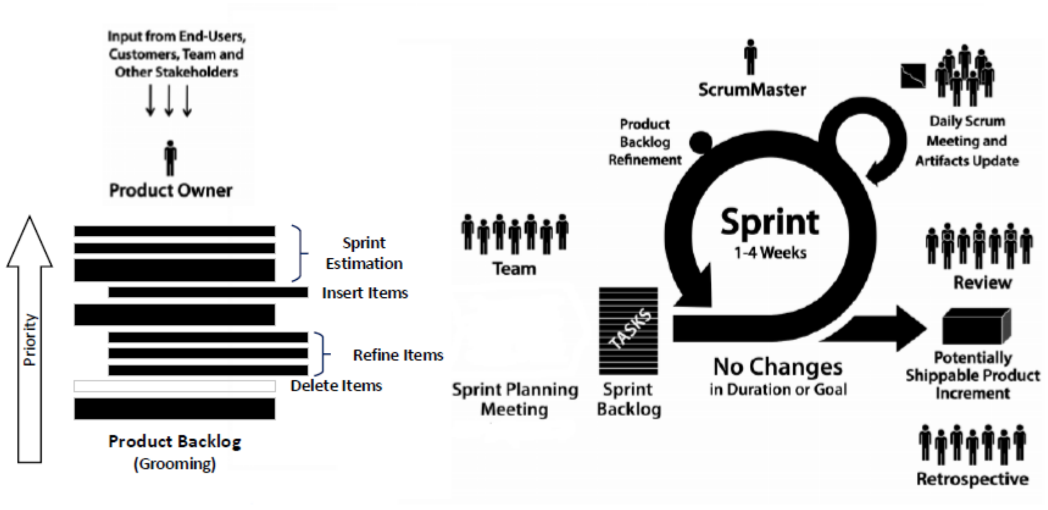

Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Priority
Sprint Estimation
Insert Items
Refine Items
Delete Items
Product Backlog (Grooming)
Team
Sprint Planning Meeting
TASKS
Sprint Backlog
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
No Changes in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

**Figure 2.5:** Scrum Framework adapted from Sutherland and Schwaber (2014)

which involves the entire scrum team. During this stage, the team selects the features that will be implemented and assign sub-tasks to developers. During development, the team will synchronise activities through a 15-minute time-boxed daily scrum meeting. At the end of a sprint, the scrum team and stakeholders will hold an inspect-and-adapt activity known as sprint review. Additionally, the SM will organise a meeting to conduct a continuous process improvement using a sprint retrospective meeting Rubin (2012). The whole cycle will be repeated again on the next sprint planning session. Table 2.2 summarises the *activities* of the Scrum framework and the roles that are involved.

| **Activities** | **Roles** |
|---|---|
| Backlog Grooming | PO |
| Sprint Planning | PO, Development Team, SM |
| Daily Scrum | Development Team |
| Sprint Review | PO, Development Team, SM |
| Sprint Retrospective | PO, Development Team, SM |

**Table 2.2:** Scrum Activities and Roles Involved

## 2.5 Agile in Embedded safety-critical

### Introduction

Agile has been the subject of safety and critical domain in recent years. Emerging medical devices are highly relying on embedded software that runs on a specific platform in real-time. The development of embedded software is different from ordinary software development due to the hardware-software dependency. Previous literature reviews discussed the challenges of bringing agile practices to embedded software developments in general. In this research, a systematic review was conducted to investigate the challenges related to agile implementation in embedded safety-critical domains (IRQ.1). Additionally, the SR investigated agile practices that have been used and the way in which the practices have been implemented in embedded medical and other embedded safety-critical domains (IRQ.2). The following subsection presents the systematic review conducted.

### 2.5.1 Systematic Review

A systematic review (SR) is a "means of aggregating knowledge about a software engineering topic or research question" Kitchenham et al. (2009). The aim of a SR is to "support the development of evidence-based guidelines for practitioners" and "provide appropriate software engineering solutions in a specific context" Kitchenham et al. (2009). In this research, a *light* version of SR, which is suggested by Kitchenham and Charters (2007), has been performed. This version of a SR is called *light* because it is manageable for a researcher (student) to perform the review with the guidance of supervisors. Figure 2.6 shows the phases involved when conducting the SR. These phases are *planning the review*, *conducting the review* and *reporting the review.*

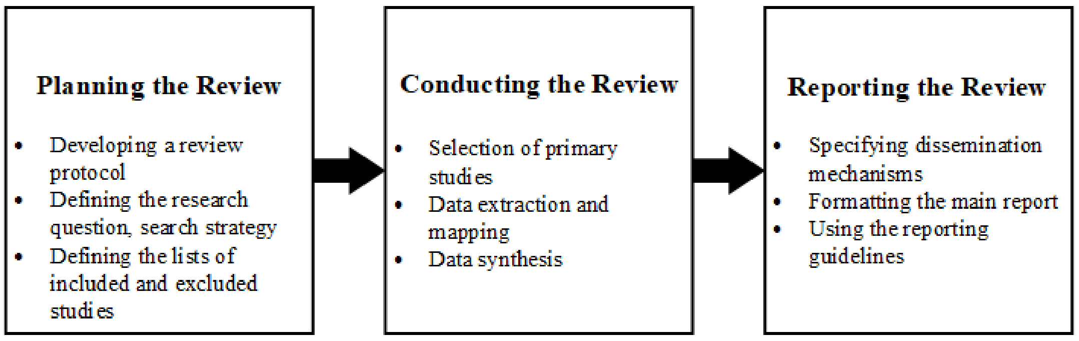

Planning the Review
Developing a review protocol
Defining the research question, search strategy
Defining the lists of included and excluded studies
Conducting the Review
Selection of primary studies
Data extraction and mapping
Data synthesis
Reporting the Review
Specifying dissemination mechanisms
Formatting the main report
Using the reporting guidelines

**Figure 2.6:** The SR Process(Kitchenham and Charters (2007))

### 2.5.2 Planning the Review

In the planning phase, a "pre-review activities" will be conducted. These are *defining a review protocol, research questions, defining the search strategy* and *inclusion and exclusion criteria.* In this research, a review protocol has been defined by the researcher and reviewed by supervisors with the guidance of Kitchenham and Charters (2007). The review protocol defines the basic review procedures, selected digital libraries, search strings, inclusion and exclusion criteria and data extraction procedure. The review protocol is shown in Appendix B.

After defining the review protocol, the planning phase will also involve defining the research questions. The research questions that are part of the SR have been designed to focus on the initial research questions, IRQ.1 and IRQ.2. The existing literature covers agile usage and challenges in the safety-critical domain and embedded system themes separately. The SR that was performed focused on agile usage in embedded safety-critical domains. The research questions that have been defined are:

IRQ.1 What are the challenges related to agile implementation in embedded safety-critical software development?

IRQ.2 What agile practices have been used, and how are the practices implemented in embedded safety-critical software development?

The search strategy which is defined in the review protocol includes search strings and the selection of digital libraries that will be used to conduct the search process. The search have been selected and arranged to address the research questions. Keywords were derived from common and emerging agile method names according to Abrahamsson et al. (2017) and VersionOne.Inc (2017) which was reported at the time of conducting the review. Additionally, keywords were also derived from embedded system and safety critical terminologies that were part of the research question. The following set of strings have been used for the search process.

```
("agile" OR "scrum" OR "extreme programming" OR "test driven development" OR "lean"
OR "DevOps") AND ("embedded system" OR "embedded software" OR "hardware development"
OR "medical" OR "safety-critical") AND  ("challenge" OR "requirement")
```

Additionally, we have applied the snowballing technique to avoid missing any relevant studies Wohlin (2014). After defining the search strings, the following digital libraries have been selected for the search process.

| **Digital Libraries** |
|---|
| IEEE Xplore |
| ACM Digital library |
| Google Scholar |
| ScienceDirect |
| SpringerLink |

**Table 2.3:** Digital Libraries Selected

The review protocol also defines a set of inclusion and exclusion criteria. These are summarised as follow:

- Inclusion Criteria
  - Studies on agile implementation for embedded software and embedded system development.
  - Studies on agile implementation for embedded safety-critical domains.
  - Studies that are published between 2010 and 2020.

- Studies that are written in the English language.

- Exclusion criteria
  - Studies discussing agile implementation for generic software development (non-embedded system development).
  - Studies that are not in the embedded safety-critical domains.
  - Studies that are not written in English language.

After defining the inclusion and exclusion criteria, a data extraction template has been defined on the tabulated format on a spreadsheet to manage the data extraction. The data extraction template should be tabulated in a manner consistent with the review questions. The template was tabulated to gather data on the challenges related to agile implementation in embedded safety-critical software development, IRQ.1, agile practices that have been used and the way in which the practices have been implemented, IRQ.2. The template includes entries such as ID, Source, Year, Author/Title, Domain, Study Type, Challenges, Agile Practices, and Summary. The contents of the data extraction template and a short description of the entries are shown in the Appendix B.

### 2.5.3 Conducting the Review

#### Selection of Primary Study

The search process was performed applying the search strings on each digital library. All search results from each database have been recorded. The first screening was performed based on *title*, *abstract* and *conclusion*. This has resulted in a total of 496 studies. In addition to the spreadsheet, the organisation of the results has been managed using Mendeley[1], an application that is used for managing and sharing research studies. Studies

[1]https://www.mendeley.com

from the first screening have been imported to Mendeley, and individual studies have been analysed based on the inclusion and exclusion criteria to further screen relevant studies.

The 496 studies have been filtered further by detailed reading, removing duplicates of similar studies published on different databases and publications by the same author reporting the progress of a project published in a short period of time. The second screening results in a total of 43 relevant studies. Figure 2.7 shows the number of selected studies from each digital library after the first and second screening stages.

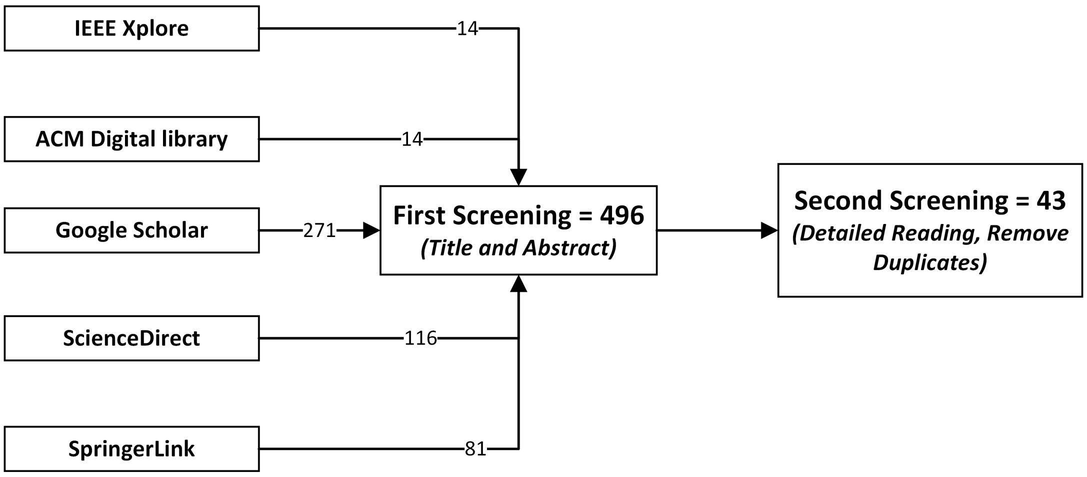


**Figure 2.7:** Screening Results

### Data Extraction

Having filtered the primary studies, the data from individual studies have been recorded on the pre-defined data extraction template. The "light" version of the SR process suggests that the extraction process has to be performed by the researcher(student) with the supervision and cross-checking of supervisors. This involves applying a “test-retest process”, that involves performing a “second extraction from a random selection of primary studies to check data extraction consistency” Kitchenham and Charters (2007). The extraction process of this SR has been conducted with the supervision of supervisors and applying a test-retest process.

After the data extraction process, the selected studies have been categorised based on the empirical study types suggested by Tonella et al. (2007). The study types include *experimental*, *observational*, *experience report*, *case study* and *systematic review.* We have also included additional categories, *Authors Opinion* for studies that report the opinion of the authors. Studies that didn't specify the methodology clearly are categorised under *Unclear* category. Table 2.4 shows the study types and description of each category.

| **Study Type** | **Description** |
|---|---|
| **Experiment** | A controlled study to observe and isolate the effect and factors involved in it. Factors other than the study that may influence the outcome should be controlled. |
| **Observational Study** | A study that gathers observations to connect factors in a non controlled manner requiring observation. Such a study often takes the form of a survey, in which random sampling is applied to select a population of cases to be observed. |
| **Experience Report** | Analysis of one case without controlling the context. The goal is to show the superiority of the proposed technology. On the other hand, setup, data collection and analysis are not discussed in detail. |
| **Case Study** | Analysis of one case and setup, collection and analysis of data will be discussed. |
| **Systematic Review** | The evaluation of all research studied in the past relevant to a topic of interest. |
| **Authors opinion** | The research present the opinion of the author or new proposed ideas that are not evaluated through empirical evidence. |
| **Unclear** | Study that didn't specify the methodology clearly or that cannot be categories under any of the other study types. |

**Table 2.4:** Empirical Study Type Adapted from Tonella et al. (2007)

### Data Synthesis

After the data extraction, the next activity of the SR process involves *data synthesis*. Data synthesis is the process of "collating and summarising the results of the included primary studies" Kitchenham and Charters (2007). Initially, the primary studies have been analysed based on the number of frequency of each study type. In the following subsections, the general overview of the studies will be presented.

### Overview of the Studies

Table 2.5 shows the categorisation of the studies under each study types. The majority of the studies are case studies and experience reports.

Previous systematic reviews have also been identified. Study P24, which also addressees the previous review P36, report the result of a review that includes agile implementation with respect to embedded software, hardware and integrated circuit. This review concludes that most of the previous reports are case studies and experience reports, and there is a lack of rigorous empirical research on the actual benefits of agile methods in embedded domain. Study P33 performs a mapping of the principles of the agile manifesto to embedded system development.

| **Study Type** | **Papers** |
|---|---|
| Case Studies | P3 P4 P7 P8 P13 P15 P16 P20 P21 P22 P23 P25 P26 P29 P30 P32 P34 P35 P37 P40 |
| Experience Report | P1 P5 P6 P9 P17 P19 P28 |
| Literature Review & Survey | P2 P10 P14 P24 P33 P36 |
| Authors Opinion | P12 P31 P39 P41 P43 |
| Unclear | P18 P27 P38 |
| Experiment | P11 P42 |

**Table 2.5:** Studies in Category

Studies from different embedded safety-critical domains have also been identified. Some of the domains are medical & healthcare (P19, P26 & P30) automotive (P8, P13 P20 & P22) aircraft/avionics (P2, P15 & P11), consumer products (P3, P6, P20 P23) and general safety-critical (P9, P15 & P30).

### 2.5.4 Reporting the Review (Dissemination)

The final step of the SR process involves reporting the review. Kitchenham and Charters (2007) states that "it is important to communicate the results of a systematic review effectively". The result can be reported in peer-reviewed academic journals and/or conferences. The result of the SR of this study has been published in peer-reviewed conference paper Demissie et al. (2018). The following subsection will present how the data collected from the SR address the research questions.

### 2.5.5 Addressing the Research Questions

This section discusses how the data extracted from the selected studies address the research questions. The synthesised overview of the challenges related to agile implementation in embedded safety-critical software development, IRQ.1 and agile practices that have been used and the manner in which each practice have been implemented, IRQ.2, will be presented in the following subsections.

#### Addressing IRQ.1

*IRQ.1 - What are the challenges related to agile implementation in embedded safety-critical software development?*

This research question aims to investigate the challenges related to agile implementation in embedded safety-critical software development. The analysis of the extracted data revealed challenges related to agile implementation in embedded safety-critical domains. The challenges that have been identified are categorised into the following groups:

- **Multiple Stakeholder Communication**: studies that describe challenges related to the specialised domain knowledge and long communication chain between diverse stakeholders.
- **Hardware Development**: studies that describe the challenges related to hardware development. These studies reported factors such as long lead-time of hardware development and hardware-software dependency.
- **Regulatory Compliance**: studies that describe challenges related to complex

regulatory compliance.

- **Tool Support**: Studies that describe the challenge related to tools and availability of tools support.

### Multiple Stakeholder Communication Challenge

This group of studies reported the communication challenge that occurred due to the diversified members involved. P1 reported the result of semi-structured interviews conducted with engineers working in a large avionics company. The study reported that diverse teams within the overall project had their own pace of completing tasks. Deadlines and milestones were defined in the contracts for the whole project, but individual teams choose their own development life-cycles within this framework, creating a "silo effect". The study also reported that the definition of "what finished is" was separate between the team that produce the circuits and the firmware team that bring the circuit to life. Additionally, the software team were working in the software plan and hardware team, and firmware people were working in the firmware plan. The lack of visibility and late-stage integration occurred because each team was focusing on its own tasks, and lost the visibility of changes that were occurring elsewhere in the project.

The need for further requirements elaboration and additional scope was reported to force changing the requirements. Such changes were reported to affect the coordination between teams and impact the project's system component. The study suggested that working closely with system, hardware and firmware teams when defining requirements and integration will help to have a better schedule and sprints. The research suggested that applying agile software development in safety critical "require the development of tools and methods" but, it didn't provide details on how the methods should be used and the usage of tools. The study concluded that there is a need to understand how agile software development can be scaled in complex systems engineering projects with multiple development teams that include software and hardware members.

P3 reports the challenge related to geo separated and multidisciplinary teams in collaboration and integration. The study also reports the challenges related to the effort required to collaborate user stories which span across different teams of cross geographical zones. P4 reports the results of a case study with nine practitioners at a large-scale embedded systems company. The study reported that interviewees mentioned communication within cross-functional teams was a challenge. The challenge was as the result of a customer representative not being sufficiently available and involved in the development team. It was suggested that agile practices would help to clarify requirement coverage and the degree to which the customer expectations are met. The study also reported that collaboration between diverse members will help in developing requirements of higher, clearer, unambiguous quality. This will result in higher software quality (fewer errors), as well as, less waste due to rework since issues will be resolved while discussing the requirements. The research was limited by the set of agile practices that were adopted by the case company and calls for further research with additional companies.

A report by study P7 stated that communication of requirements changes was a challenge at one of the companies involved. The study also reported that writing or revising user stories or acceptance test cases by the business roles was difficult due to the limited technical knowledge of the customer to discuss quality requirements. Additionally, there was a lack of communication of implemented changes between roles. The study highlighted that the case company's product line software was running on several different hardware versions and "a test case may fail due to variability in the underlying hardware rather than that the requirement is not fulfilled". A change might be applied to fix the test case and also rewrite the requirements without communicating the changes. This was reported to make the tracing and managing of test cases difficult. The study concluded that further studies are required to investigate requirements format and improve the coordination and effectiveness of agile requirements practices.

P8 reports the result of a study that aims to investigate the applicability of agile methods to the software development at Volvo Car Cooperation (VCC). The study reported that the introduction of agile methods was hindered by individualism and long communication chains. The study stated that information was passing through different channels. For example, to clarify unclear requirements, “information has to go through the software responsible, to the architects, to the electrical department and so on”. Additionally, the study also stated the lack of domain knowledge between diverse teams such as for example, hardware and electronics knowledge than software knowledge resulted in the creation of vague requirements.

Study P11 addresses the challenge of team-based communication when five distributed teams that are working on different interfaces of an aircraft cockpit display system. The study reported that communication between the team was poor because different members of the team have been modifying the specification of interface requirements independently and that results in problems and delays during the products hardware-software integration. The research conducted was in academic setting that mimics the execution of an aircraft embedded software project and lacks real industrial environment.

In P14, the author discussed the challenges that need to be addressed in mass-produced embedded systems. Scaling the system engineering activities of this domain requires cross-functional teams to collaborate as “components in a large-scale system are technically very difficult and interdependent and require years of experience to be fully understood by developers.” The research calls for broadening research on scaling agile to address the challenge.

Study P16 highlighted the importance of close cooperation and teamwork between developers and testers to improve requirement engineering practices and avoid a rework and technically infeasible requirements at an early stage. Based on interviews with different roles in the six companies, the study recommended that “testers’ reviews of requirements as a good practice that enhances both the communication and the quality of the requirements, thereby resulting in better alignment of the testing effort”. This will result in early identification of problems with the test specification and avoiding more expensive problems in the future.

P20 investigated the effect of interaction speed and business goals in a large scale company developing in-house embedded software. The study reported that one of the key factors affecting interaction was knowledge unavailability. The team without the required knowledge of a particular feature will need to interact with an external expert. The study reported that such an expert might not be available, and this will force the team to make assumptions that lead to redoing most of the work. The other reported root factor, affecting the interaction speed, was the availability of unclear requirements. The study highlighted that to clarify unclear requirements, and the team was also affected by long waiting time and the continuous interaction for clarification.

Study P23 investigated requirement engineering challenges related to a product with team composition of software and hardware of automotive companies. The study reported communication-related challenges such as creating a shared understanding of value and building and maintaining system understanding. Regarding shared understanding, the study stated that writing meaningful user stories and high-level user stories could help to communicate value early and "offer a unique opportunity to bridge distances between customer and developer". Additionally, the study also reported that "requirements need to be discussed with and communicated to other stakeholders within or outside the organisation, delaying feedback". Building and maintaining system understanding between system managers and agile teams was also reported to be challenging. The study stated that agile teams could add or change backlog items in collaboration with the product owner without notifying the systems engineers about the changes. The study recommended that increased gate-keeping and backlog grooming by the agile team and system engineers will be required to avoid issues during system integration and testing.

In P25, the result of a case study conducted at Ericsson involving a cross-functional team was reported. The study reported that inefficient communication happened between team members as a result of "over specialisation" that refers to team members focusing on their own task and not having adequate knowledge about the overall product. The interviewees from the case study highlighted that problems with knowledge sharing create a challenge for communicating technical dependency and calls for an effective way of communication across teams.

Study P30 reported the result of three case studies on the adaptation of agile practices into embedded system development involving hardware development. The author stated that “agile methods offer benefits to embedded system development similar to benefits of software benefits, but the methods need to be tailored due to the special characteristics of embedded system development.” One of the special characteristics that have been reported is “the different knowledge between developers, which e.g., required tailoring the planning meeting according to the disciplines.”

In P34, the challenges of automotive requirements engineering (RE) have been investigated with respect to communication and organisation structure. The study reported that high-level requirements are often vague and abstract. On the other hand, lower-level requirements, are very detailed and technical, and in many cases, there is a gap. “It can happen that engineers add details to the requirements or correct them, without communicating it”. Additionally, the study reported that there is a lack of common interdisciplinary understanding because communication between the multiple disciplines involved in the development has been challenging.

### Hardware Development

Study P14 reports that companies within the embedded systems domain struggle with the alignment of hardware and software development cycles. Another study, P24, state the difficulty of managing hardware iterations which results in different paces of software and hardware development. Study P30 also states that the slower nature of hardware development affects the implementation of agile methods in embedded systems development.

The systematic review reported by study P36 stated that hardware dependencies affect applying agile methods to embedded software development because of the long development cycles.

In study P37, the experiences and best practices when combining Lean and agile hardware development have been reported. The study stated that “one of the differences between hardware and software development is that you can split the software into (almost) arbitrarily sized bites but not the hardware iterations”. Additionally, development errors may arise from the interaction between hardware and software. But "those won’t be known / can’t be resolved until all software is completed."

In study P38, the author stated that hardware is not available during design time and “software testing is often considered to be impossible.” As a result, testing will mostly be postponed until after hardware development. The author highlighted that as testing is the last phase in the process, it might be shortened when the deadline is nearing.

### Regulatory Compliance

The availability of standards and regulation process is reported to bring difficulty in implementing agile in regulated domains. Studies such as P2, P15, P39 and P40 reported this challenge. A survey conducted by P2, in the European avionics industry reported that “verification and certification constitute a large portion of the total costs of development (estimated 40%)”.

In P15, an industrial case study to develop a high-integrity fire and gas detection system using Scrum according to the IEC61508 standard have been reported. The report states that “the inherent quality assurance mechanisms in Scrum are not sufficient to meet the demands imposed by the IEC61508 standard.” The study analysed the standard and with the consultation of an independent assessor, customised the Scrum process by adding additional tasks for a team-internal QA role.

Study P39 analyses two standards, IEC 62304 and IEC 82304-1 and reports on adopting DevOps methods in a tightly regulated software development of medical devices. The study states that such standards require special attention in relation to continuous integration and make it difficult using continuous deployment after deployment to the customer. The study calls for new tools and methods to be specifically developed for using DevOps in regulated software development.

Study P40 reported that the development of airborne systems following standard DO-178B has "milestones and project progress connected to audits (on system design or software) or formal documents (a document is issued that is required at a certain stage in the process)." As a result, such compliance to standards shifts focuses away from working software.

#### Tool Support

The importance of proper tools to provide a high level of automation have been reported in study P2. Study P6 report the challenges of DevOps adoption in embedded systems using a multiple-case study. According to the study "the embedded systems domain, especially in critical systems, there is a lack of technology that would allow new software features to be automatically deployed repeatedly and reliably on a continuous basis."

#### Summary

In this subsection, the result of the SR to address the research question, IRQ.1, has been reported. Table 2.6 summarises the challenges that have been identified and the number of studies reporting the challenges. The majority of the studies reported the challenge of *multiple stakeholder communication*. The challenge related to *hardware development* has also been reported in some of the studies. This is followed by the challenge of *regulatory compliance* that has been reported in four studies. Two studies reported the challenge of *tool support*.

The key finding related to the IRQ.1 is that the challenge of multiple stakeholder communication was reported in most of the studies. The studies reported that the embedded safety critical domains were affected by the lack of collaboration between diverse members and agile methods need to be tailored to address this challenge.

| **Challenge** | **Studies** |
|---|---|
| Multiple Stakeholder Communication | P1 P3 P4 P7 P8 P11 P14 P16 P20 P23 P25 P30 P34 |
| Hardware Development | P14 P24 P30 P36 P37 P38 |
| Regulatory Compliance | P2 P15 P39 P40 |
| Tool Support | P2 P6 |

**Table 2.6:** Challenges in the Studies

### Addressing IRQ.2

*IRQ.2 - What agile practices have been used, and how are the practices implemented in embedded safety-critical software development?*

This research question aims at investigating agile practices that have been used in embedded safety-critical software development and the way in which the practices are implemented. The majority of studies report the implementation of a combination of agile practices. In studies P2, P3 and P19, the implementation of practices from *Scrum* and *XP* has been reported. In P2 the result of a survey conducted among European avionics industry has been reported. The study extracted practices such as *Test-Driven Development (TDD), Coding Standards, Design Improvement/Refactoring* and *Planning Game* to be usable in an avionics development process that is governed by DO-178. The study concluded that "agile methods may promise to resolve some of the specific challenges in the avionics domain, but there is still a clear need for more research and industrial experimentation to verify the applicability and to demonstrate improvement effect." In P3, the author shares several changes that take place when the transition takes place from V-mode methodology to Scrum in Embedded Software Testing Industries. The study states that the transition challenges the testers to think outside of the box as customer-centric than relying on manual procedures to perform day-to-day testing tasks. On the other hand, study P19 report the implementation of Scrum and XP practices within a medical device software development organisation based in Ireland. Practices such as *On-Site Customer, User Stories and self Organising Teams* have been reported.

In some reports, the Scrum framework has been extended in different variations. For example, P15 extends Scrum with additional XP techniques to develop safety-critical software and have the software certified according to the IEC 61508 standard. The modified Scrum, known as SafeScrum, implements two backlogs for functional requirements and safety requirements. The extension also includes the usage of agile practices such as *Test-Driven Development, Daily Stand-ups, and Continuous Integration.* Study P41 propose the Scrum variant for cyber-physical systems (CPS) known as Scrum-CPS. This report proposes two sprints, a Design Sprint and a hardware sprint that synchronises using the concept of Agile Release Train (ART).

In addition to combining and extending existing agile practices, some studies such as P5, P11 and P42 report the combination of agile practices with Model-Based Development (MBD). In P5 agile practices from Scrum and XP have been combined with MBD to develop a spatial real-time embedded system. Five Scrum teams develop and integrate two different Cockpit Display Systems using the safety-critical Application Development Environment (SCADE) Suite. The study reports the implementation of practices such as *Sprint Planning Meetings, Daily Meetings, Sprint Reviews, Sprint Retrospectives, User Stories and Pair Programming (PP).* Regarding PP, the study states that "this applied technique also helped to speed up the delivery and communication among members."

Study P11 report the implementation of aircraft cockpit display system using Scrum practices combined with MBD. In P42 the implementation of Scrum for the integration, and management of nanosatellite event Simulator is reported. The study combined Scrum with the software Safety-Critical Application Development Environment (SCADE), which is a MBD tool that helps to simulate and generate code automatically from the state machine representing satellite operational phases.

In addition to Scrum and XP, some studies report the implementation of Test Driven Development (TDD), Lean, DevOps and Behaviour-Driven Development (BDD). Studies such as P2, P15, P18, P26 and P38 report the implementation of TDD. Study P2 investigated agile methods for Avionics software development. The study stated that TDD with other practices from Scrum needs to extend the existing testing activity.

Study P9 reports the result of an investigation on the expectations and challenges from scaling Agile in organisations dealing with mechatronics development. The study states that TDD is applied by one-third of the respondents. On the other hand, study P15 reports the importance of TDD to establish high test coverage when developing safety-critical software according to the IEC 61508 standard. Additionally, studies such as P18, P26 and P38 reported implementation and benefit of TDD.

Study P37 propose a Lean-Agile Framework for hardware development using practices such as Two-Level Rolling Planning, Cadence, Synchronisation, and Key Decision Points. In study P43, on the other hand, the usefulness of the Lean method has been investigated for developing medical devices. The study identifies key influences on the software development life-cycle (SDLC) and suggests that the Lean software development model "can be employed while not affecting regulatory compliance."

Regarding *DevOps*, studies P6 and P39 report the implementation of DevOps in embedded and regulated software development domains. In P6, the challenges for DevOps adoption in embedded systems have been reported. The study states that cross-functional teams and continuous integration practices still need improvement. Additionally, in the embedded systems domain practice such as Acceptance Tests(ATs) "takes several months after the completion of system development". The study calls for future research to tackle the challenges. Study P39, on the other hand, investigated adopting DevOps in tightly regulated software development using two standards, IEC 62304 and IEC 82304-1. The study calls for “further analysis of medical device software and DevOps.”

The implementation of BDD has been reported in study P32. The study reported the integration of hardware-software (HW-SW) co-design with the BDD to allow the development teams to work concurrently without delay by other teams. The study stated that BDD provides the “ability to describe the behaviour of the software as executable user stories in HW-SW) co-design environment.” Table 2.7 summarises the list of agile practices and corresponding studies that report the implementation of the practices. Table 2.8 summarises the combination of agile methods that have been identified.

| **Agile Practices** | **Studies** |
|---|---|
| Test-Driven Development | P2 P15 P18 P26 P38 |
| Continuous Integration | P15 P18 P26 P40 |
| Daily Scrum (Stand-up) | P1 P8 P15 P18 |
| Sprint Retrospective | P5 P8 P18 |
| User Stories | P4 P5 P19 |
| Acceptance Criteria/Testing | P4 P6 P26 |
| Pair Programming | P1 P5 P8 |
| Cross-Functional Teams | P4 P6 P8 |
| Cadence | P30 |
| On-Site Customer | P19 |
| Coding Standards | P2 |
| Two-level planning | P37 |
| BDD | P32 |

**Table 2.7:** Agile Practices Identifies

| Methods | Studies | Modifications & Suggestions |
|---|---|---|
| Scrum | P3 | The methods need to extend testing activities by having more acceptance testing |
| | P12 | Scrum was adopted to be linked with the Co-Design for the development of software and hardware. |
| | P29 | Scrum was used in pilot projects. |
| | P30 | Dividing the work into iterations, especially on the hardware side was experienced difficult, and all the tasks were not completed in one iteration. |
| | P35 | |
| Scrum and XP | P9 | Adopted a combination of Scrum practices combined with XP for open source safety-critical software. |
| | P19 | |
| Scrum and MBD | P5 | Development of a spatial real-time embedded system within using Scrum and SCADE system tools. |
| | P11 | |
| | P42 | Scrum combined with model-based design. |
| Scrum-CPS | P41 | Proposed two sprints for design and hardware that synchronises using the concept of Agile Release Train (ART) |
| SafeScrum | P15 | Extensions to make Scrum applicable to the development of safety-critical software using two backlogs for functional and safety requirements |
| Lean | P43 | |
| DevOps | P6 | Needs to address poor communication between developers and infrastructure owners using DevOps. |
| | P39 | |

**Table 2.8:** Agile Methods Combination

### 2.5.6 Summary of the SR

In this research, a SR has been conducted to answer the research questions, IRQ.1 and IRQ.2. The investigation on the challenges related to agile implementation in embedded safety-critical domains have shown that challenges such as multiple stakeholder communication, hardware development, regulatory compliance and tool support have been reported in the studies.

In addition to identifying the challenges of agile usage in embedded safety-critical domains, the review has also investigated agile practices that have been reported to be used in embedded safety-critical domains (IRQ.2). Most of the studies reported the implementation of more than one agile practice. The combination of Scrum and XP practices have been reported in most of the studies. Practices such as Acceptance Testing, Cadence, Two-level planning and BDD have also been reported.

The review has also investigated the way in which individual practices have been implemented or modified. Different agile practices have been modified or combined with other development technologies such as model-driven development. Scrum has been modified to address functional and safety requirements (SafeScrum) and hardware-software designs (Scrum-CPS).

As shown in Table 2.7, the majority of agile practices that have been reported in the SR are drawn from agile methods such Scrum and XP. Additionally, practices from ATDD and BDD are also reported. As such practices were used in embedded safety critical domains, they are considered suitable candidate for embedded medical software development. In the following subsections, a short summary of such candidate agile practices that have been reported in more than one study will be presented.

## 2.5.7 Suitable Agile Practices

### TDD

TDD is a practice that focuses on writing automated tests before the production code that specifies and validates what the code will do. “For every tiny bit of functionality in the production code, you first develop a test that specifies and validates what the code will do. You then produce exactly as much code as will enable that test to pass.” Janzen and Saiedian (2005).

The steps to implement TDD is based on the red-green-refactor cycle defined by Beck (2003). The steps are:

1. Add a unit test;
2. Run all tests and see the new one added in step 1 fail (Red).

3. Make a little change to satisfy the new change.

4. Run all tests and see them all succeed (Green).

5. Refactor to remove duplication.

TDD helps users write better code. This is because testable code is written by default Beck (2003), Jeffries and Melnik (2007). TDD is reported to encourage programmers to learn quickly, communicate more clearly, and seek out constructive feedback Beck (2003). This will make the practice suitable to assist the communication between developers. TDD also helps software developers produce working, high-quality code that's maintainable and, most of all, reliable Karlesky et al. (2006).

### Continuous Integration

According to Fowler and Foemmel (2006), "continuous integration is a software development practice that requires team members to integrate their work frequently." Code written locally by developers will be merged into a common shared source repository called source control. Developers check out the latest common version of files and check in any changes they make. When a change is applied to a source control repository by a developer, it will be built automatically by a continuous integration server.

In addition to the server, continuous integration also needs a comprehensive automated test suite. Humble and Farley (2010) state that "it's essential to have some level of automated testing to provide confidence that your application is actually working." The automated tests are used to check if the resulting build made by developer actually works. Continuous integration gives the confidence that the latest build version is always ready for production deployment. This is because all code is tested and only merged only when all tests are passed.

### Daily Scrum (Stand-up)

The Daily Scrum is a 15-minute time-boxed event for the development team to synchronise activities and create a plan for the next 24 hours. This is done by inspecting the work since the last Daily Scrum and forecasting the work that could be done before the next one. The Daily Scrum is held at the same time and place each day to reduce complexity. The Daily Scrum helps the team to "inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog" Schwaber and Sutherland (2017). This meeting helps to improve communications and promote quick decision making. Additionally, the development team's level of knowledge will also be improved.

### Sprint Retrospective

A Sprint Retrospective is a continuous improvement activity that is held prior to the next Sprint. The objective of this activity is to inspect the previous Sprint with regards to people, relationships, process, and tools and plan for improvements on the next Sprint. "The Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of 'Done'." Schwaber and Sutherland (2017)

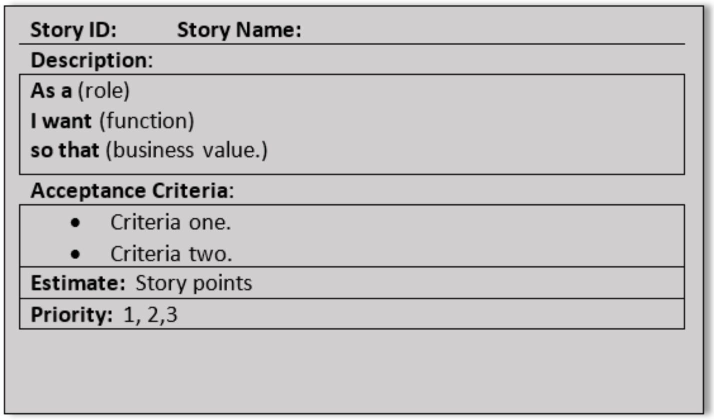
### User Stories

User stories are a simplified description of features that are broken down into smaller pieces. "All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality" Cohn (2004). User stories are usually written on sticky notes and arranged on walls or tables to facilitate visualisation and planning. Jeffries (2001) defines the common model to capture the components of User Stories called Three C's. The model involves Card, Conversation and Confirmation. The card represents a token representing the requirement with notes written reflecting priority and cost. The conversation is used to show that details come with the exchange and opinions of the customers and the product owner. The Confirmation represents the ATs of the User story.

User Stories are written in different formats or templates. The two common formats are the *role-feature-reason* format and *Given-When-Then* format.

- Role-Feature-Reason - this format is also known as *Connextra template*. The story is written in the following format:
  *As a (role), I want (function) so that (business value.)*

  The layout of this format is shown in Figure 2.8.
- Given-When-Then - this format is commonly used to write AT with User Stories. The format is based on human-understandable cause-and-effect clauses known as *Gherkin scenarios*. Gherkin is a domain-specific language that is adopted from the Cucumber syntax that defines a series of steps using *Given/When/Then* statements Wynne et al. (2017). A typical user story description based on this format is shown in Figure 2.9



**Story ID:** **Story Name:**

**Description:**

**As a** (role)
**I want** (function)
**so that** (business value.)

**Acceptance Criteria:**

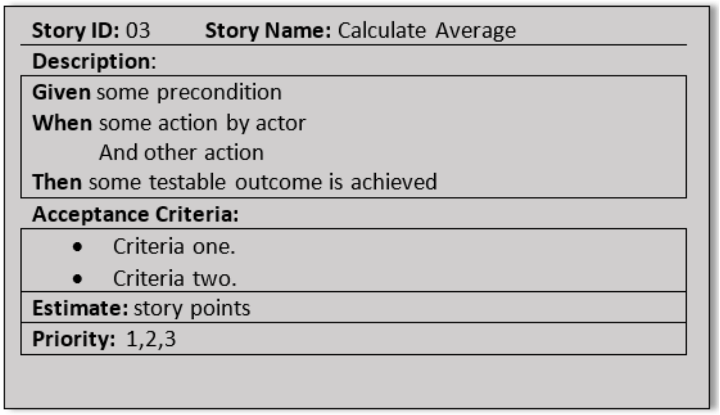
- Criteria one.
- Criteria two.

**Estimate:** Story points

**Priority:** 1, 2,3

**Figure 2.8:** Role-Feature-Reason

The user story description using both formats have an estimation and prioritisation. Estimation refers to the overall effort that will be required to implement the user story. It is conducted using story points, numbers drawn from a pool of numbers of a set size.



**Story ID:** 03 **Story Name:** Calculate Average

**Description:**

**Given** some precondition
**When** some action by actor
And other action
**Then** some testable outcome is achieved

**Acceptance Criteria:**

- Criteria one.
- Criteria two.

**Estimate:** story points

**Priority:** 1,2,3

**Figure 2.9:** Given-When-Then Format

For example using Fibonacci-like sequence a user story could have 1,2,3,5,8,13,20,40 or 100 story points. The reason for using a Fibonacci-like sequence is to encourage stories to be estimated relatively. For example, assuming user story-1 has been assigned two story points, if user story-2 requires about twice the effort of user story-1, it probably can be a five story points Cohn (2004).

Additionally, the prioritisation of user stories is used to decide stories with the highest value to be implemented first. The common way of prioritising stories is using numerical values (1-3). User stories classed as 1 are critical and must be delivered first. User stories with medium priorities are classed as two and stories with low priorities are classed as 3.

User stories are reported to enables teams to develop the right software and "facilitates creating a common understanding concerning the requirement" Lucassen et al. (2016).

## ATDD

ATDD is a practice that involves the collaboration of the *whole team* to discuss acceptance criteria with *examples* and then distills them into a concrete set of acceptance tests (ATs). The ATs represent the expectations of behaviour the software. ATDD involves the collaboration of the *development team*, *business (customer) team* and *testing team*. The three participants are called in different terminologies. For example, Pugh (2010) calls them "the triad" while in Gärtner (2012), they are called "the-three-amigos".

The ATDD cycle, defined by Hendrickson (2008), has four major phases. These are *discuss, distill develop and demo*. Figure 2.10 shows the ATDD cycle. In the following subsections, a short summary of each phase will be presented.

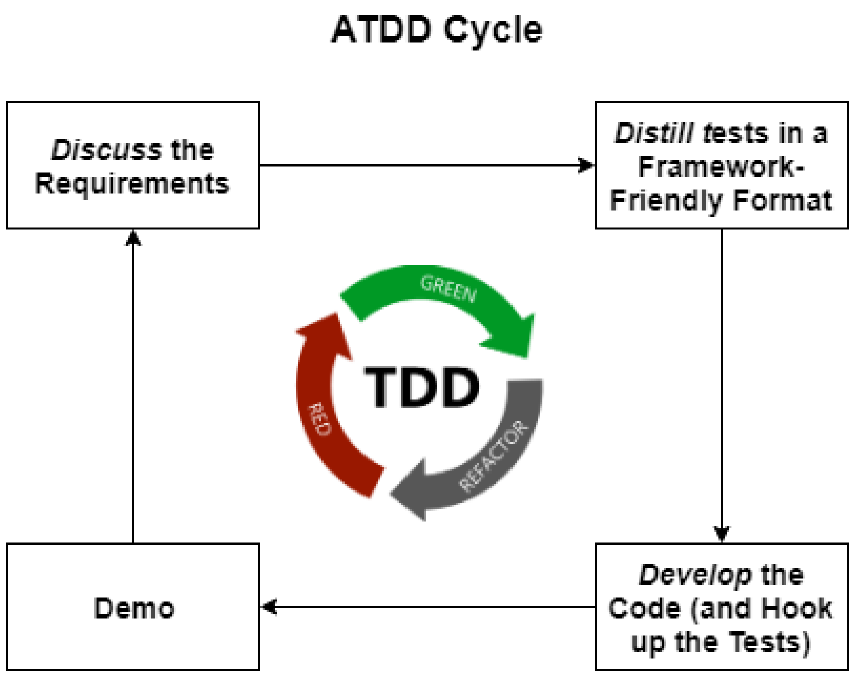

ATDD Cycle
Discuss the Requirements
Distill tests in a Framework-Friendly Format
Develop the Code (and Hook up the Tests)
Demo
TDD
GREEN
REFACTOR
RED

**Figure 2.10:** ATDD Cycle (adapted from Hendrickson (2008)

### Discuss

The first phase of the ATDD cycle involves the discussion of the user stories. This phase is also called the *specification workshop*. The discussion will be conducted between the development team, testers and business stakeholders during the *planning* phase. The discussion and collaboration will help to have a shared understanding of the project and to clarify any misunderstanding of the external view of the system and what the business stakeholder expects. Pugh (2010) stated that "the members learn from each other about the business domain and the development and testing issues." The team will discuss each user story and then write acceptance criteria for each user stories. Acceptance criteria define conditions or boundaries that a user story must satisfy so that the story will be accepted by a user. In short, acceptance criteria are "general condition of acceptance" that defines *what needs to be done* Pugh (2010). Acceptance criteria need to be written clearly without any ambiguity in a language that is understandable by all members. A common form of writing acceptance criteria is a bullet point highlighting the intent of the story where each bullet point is a condition.

After defining the acceptance criteria, the team will discuss to write acceptance tests (ATs). ATs are scenarios which are derived from acceptance criteria with *detailed specifications of acceptance* of the system behaviour. Each acceptance criteria can have one or more ATs. ATs use test cases to validate scenarios. According to IEE (1990), acceptance test cases are defined as "a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement." A particular test case might also include *subtests* that will also be executed as a sequence. "The outcome and/or final state of one subtest is the input and/or initial state of the next." (Beizer, 1995, p.3). Two common notations used to write ATs are:

- Fit Tables - Fit, stands for a framework for integrated tests, is developed by Ward Cunningham to help to get customers involved in writing acceptance tests Mugridge and Cunningham (2005).In this notation, ATs are written in tabular notation. The rows and columns of a table provide a clear and simple structure.
- Gherkin Language - a domain specific language that use plain-text language with a extra structure. The notation uses regular spoken language to describe requirements and scenarios Wynne et al. (2017). Gherkin uses keywords such as *Feature, Background, Scenario, Given, When, Then, And* and *But.*

The outcome of this phase is a set of ATs in the form of concrete expectations and examples that can be executed.

#### Distill

The next phase involves the development of test cases in a format that works with the test automation framework. Test cases that are developed previously will be turned into an *executable format* that can be loaded to a testing framework. Variety of tools such as Fitnesse, Cucumber, Robot Framework and Concordion are used to automate the test cases.

#### Develop

This phase involves hooking up the test when developing the code. The development team will write code using the *TDD* approach and execute the acceptance tests. During development, the developer might identify scenarios that were not identified earlier

#### Demo

The final phase involves demonstrating the product to the business stakeholders and Product Owner (PO) to re-validate the tests and ATs.

### Pair Programming (PP)

In his book, *Extreme Programming Explained: Embrace Change*; Kent Beck explained PP as a way of programming where "all code is written with two programmers at one machine". The two pairs can interchange roles. One of the pairs is known as the *driver*, who is in control of the keyboard and is thinking about the best way to solve the problem. The second member of the pairs, the *observer*, is observing the whole approach and look for "test cases and perform code inspections" Beck and Andres (2005). Through pairing, developers or engineers take parts in development, coding, updating and writing unit tests. This practice is reported to increase the quality of the work product and increase the knowledge of each engineer such that the total time to implement a project is lower with pair programming than without Alistair Cockburn (2000), VanderLeest and Buter (2009).

One of the main benefits of PP is knowledge sharing. Alistair Cockburn (2000) states that “knowledge is constantly being passed between partners, from tool usage tips (even the mouse) to programming language rules, design and programming idioms, and overall design skill.” According to a meta-analysis study performed by Hannay et al. (2009), the previous study on PP shows that this practice can help in achieving correctness on highly complex programming tasks.

Another study performed by Williams (2010) states that PP can be used as a form of continuous review, debugging and problem identification. The study summarised that PP has the following benefits for industrial teams:

- Knowledge management
- Enhanced learning
- Higher product quality
- Enhanced team spirit
- Improved cycle time
- Reduced product risk

### PP and Its Variants

The investigation of this practice has shown different flavours and variations:

- **Cross-Functional PP (CFPP)** James E. Hewson (2003), combines one software engineer with one hardware engineer working for the same objective. This can be the embedded project they are working on.

- **Pair Design (PD)** a more hybrid concept of PP. PD has become a common trend in User Experience (UX) design. UX design is the process of designing products that are easy to use and interact. It focuses on enhancing the experience that people have while interacting with the product. PD involves two designers working on the same problem simultaneously for the duration of the project. PD "reduces the communication overhead of a design team", and also provides "higher quality design with less documentation" Anderson and Noessel (2017). For example, a front-end developer working with a graphic designer to build the grid during the sketching session. The graphic designer and front-end developer are "made aware of each other" as they are working on the same grid.

- **Remote Pair Programming (RPP)**, a form of PP that is based on real-time collaboration tools where remote pairs collaborate using collaborative code editing tools. This form of PP is affected by "network latency, possibly from bandwidth limitations, and from having to cope with the partner's IDE configuration (e.g. colours, layout, key bindings, and so on)." Schenk et al. (2014)

- **Distributed PP (DPP)** Baheti et al. (2002), two members of the team synchronously collaborate on the same design or code from different geographical locations. Unlike RPP, that uses different collaborative tools, DPP uses dedicated collaboration tools that are distributed and replicate the files to both pairs locally and keep them in sync.

- **Mob Programming**, a software development approach where the whole team work at the same time, in the same space, and at the same computer to deliver a single work item continuously collaborating. "The team works together to do almost all the work a typical software development team tackles, such as defining stories, designing, testing, deploying software, and working with the customer" Zuill and Meadows (2016)

Table 2.9 summarises the variations of PP and the combinations of pairs that involve in the implementation.

| PP Variation | Pair Composition |
|---|---|
| Cross-Functional PP (CFPP) | Software Engineer and Hardware Engineer |
| Pair Design (PD) | Designer and Front-end Developer, or Two Designers |
| Remote Pair Programming (RPP) | Two Programmers remotely using their own real-time tools. |
| Distributed PP (DPP) | Two Programmers remotely using the same real-time tools that are located on their local machines. |
| Mob Programming | The whole team working at the same space. |

**Table 2.9:** PP Variations

#### Cross-Functional Teams

A cross-functional team is a team that is composed of members from different functional area to achieve a common goal. This practice is reported to increase the productivity, creativity and organisational learning of the development team Parker (2003).

## 2.6 Conclusion of Research Background

This chapter begins by describing the embedded systems design process and challenges of developing embedded safety-critical software. The industrial investigation which was conducted to explore the embedded medical software development process was also presented. This chapter also describes agile software development and agile methods that are reported to be mostly used in embedded safety-critical domains. In the following subsection, the research questions IRQ.1 and IRQ.2 will be revisited.

### 2.6.1 Research Questions and Objectives Revisited

IRQ.1 *What are the challenges related to agile implementation in embedded safety-critical software development domains?*

The SR was conducted to investigate the challenges related to agile usage in embedded safety-critical software development. The systematic review revealed that the challenge of multiple stakeholder communication has been reported in most of the studies.

IRQ.2 *What agile practices have been used, and how are the practices implemented in embedded safety-critical domains?*

The SR investigated suitable agile practices that have been used in embedded safety-critical software development. The review reported that most of the studies reported the implementation of a combination of agile practices. Practices from Scrum and XP have been reported in most of the studies. Additionally, practices such as acceptance testing, BDD and Synchronisation have also been identified.

Using IRQ.1 and IRQ.2 the challenges related to agile usage and agile practices that have been used in embedded safety-critical domains have been investigated. A focus has been applied, and this study will focus on addressing the challenge of multiple stakeholder communication in embedded medical software development domain. The refined research question, RRQ, will aim to address this challenge by proposing a process using a combination of suitable agile practices. This will be discussed in chapter 4. In the next chapter, the research methodology will be presented.

# Chapter 3

# Research Setting

## 3.1 Research Methodology

A research methodology is the "strategy, plan of action, process or design" behind the choice of a particular research method Crotty (1998). (Guba et al., 1994, pp.108) stated that "methodology asks the question: how can the inquirer go about finding out whatever they believe can be known?". The research onion proposed by Saunders et al. (2009) has been used as a guideline to design this research. As shown in Figure 3.1, the research onion has different layers, and the researcher has to make a systematic choice peeling away each layer.

### 3.1.1 The Research Onion

The outermost layer of the research onion describes research philosophy.

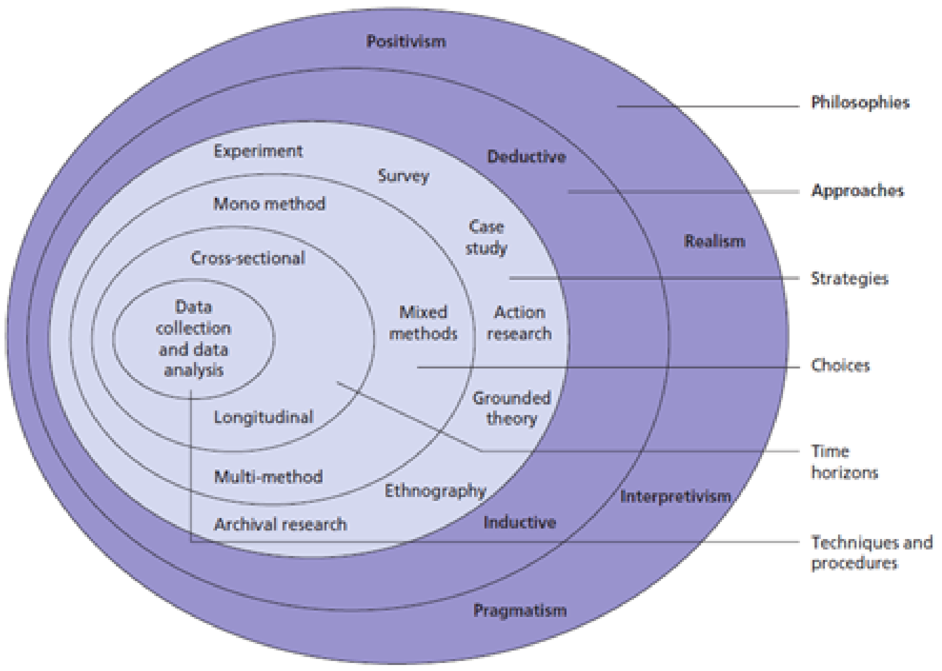


**Figure 3.1:** The Research 'Onion' Saunders et al. (2009)

## 3.1.2 Research Philosophy

According to Saunders et al. (2009), research philosophy refers to the formulation of knowledge . There are four types of research philosophies. These are positivism, interpretivism, realism and pragmatism.

## 3.1.3 Research Approach

The next layer of the research onion is research approach. It describes how the findings of the theory have been analysed and presented. Two main research approaches outlined by Saunders et al. (2009) are deduction and induction. When a deduction is used, theory and hypothesis are developed, and a research strategy is designed to test the hypothesis. On the other hand, induction data collection involves theory being developed as a result of the data analysis. A summary of some of the strategies will be presented in the following subsections.

### 3.1.4 Research Strategies, Choices and Time horizons

Peeling away research approach, the next three layers which are: research strategies, research choices and time horizons can be thought of as the main ingredient for turning the research question into a research project. The research strategy is defined as "the general plan of how the researcher will go about answering the research questions" Saunders et al. (2009). There are different research strategies from which a researcher may select. These strategies have "large overlaps among them, and hence the important consideration would be to select the most advantageous strategy for a particular research study" Yin (2003).

The research choices helps in deciding the number of data types (qualitative or quantitative) that will be used. The choices are mono method, mixed method, and multi-method. In mono-method, we will use one data type, either qualitative or quantitative. On the other hand, both data types are used in mixed method. The multi-method uses wider range of approaches. Creswell and Creswell (2017).

The time horizons defines the time frame that the project is intended for completion Saunders et al. (2009). The research onion defines two types of time horizons: cross-sectional and the longitudinal. The cross sectional time horizon is one where the data is collected at a certain point. On the other hand, the longitudinal time horizon refers to the collection of data repeatedly over an extended period. This approach is used when the research is examining change over time. The next subsection presents a short summary of some of the research strategies.

#### Experiment

This research strategy focuses on the causal links between independent and dependent variables. The investigation is conducted by controlling the "conditions experienced by participants, administer an intervention, and then test whether the intervention affects the outcome" Shadish et al. (2002). Experimental research often conducted in laboratory settings to control of sample selection, measurement of variables and control of other variables. These settings are "unlikely to be related to the real world of organisations" as the extent to which the findings can be generalised to all organisations is likely to be lower than the real world of organisations Saunders et al. (2009). Therefore, experiments are not a suitable method for this research as the focus of this study.

### Survey

Survey research includes the selection of a sample from a "well-defined population, and the data analysis techniques used to generalise from that sample" Easterbrook et al. (2008). This strategy allows the collection of a large amount of quantitative data using a questionnaire, and the data will be analysed using descriptive and inferential statistics. The data will be collected from "only a fraction of the population, that is a sample, rather than from every member of the population" Kitchenham and Pfleeger (2008). According to Easterbrook et al. (2008), survey research is suited for research which seeks to answer "what" questions. Survey research is not considered for the initial, problem investigation, phase of this research as a SR is considered to provide a more in-depth insight into the investigation.

### Grounded Theory

Grounded theory is a research strategy that aims at generation of theory, that is "grounded" in the systematically collected and analysed data Glaser and Strauss (1967). In grounded theory, data is collected, marked with codes which are then grouped and categorised to form the bases for the creation of theory. Grounded Theory is not a suitable method for this research as the focus of this study is not theory development.

## 3.2 Case Study Research

From these various strategies, this research sought to adopt the case study research strategy as the appropriate strategy for research. A case study is an "empirical inquiry about a contemporary phenomenon (e.g., a "case"), set within its real-world context-especially when the boundaries between phenomenon and context are not clearly evident" Yin (2012). The case study is a flexible design strategy with a wide variety of data collection methods, including direct observation, participant observation, interviews, focus groups, documentary sources, archival records, and physical artefacts.

Figure 3.2 shows the design process of case study research adapted for this research based on Yin and Robert (2009). The research questions, IRQ.1 and IRQ.2 and objectives OBJ.1 and OBJ.2 are the initial starting point of the study. A SR has been performed to constitute a conceptual and theoretical structure and address IRQ.1 and IRQ.2. The research proposed a solution to address multiple stakeholders communication of embedded medical and safety critical domains using a combination of suitable agile practices (RRQ) and address the objective stated at OBJ.3.

After defining the conceptual and theoretical structure, the next step includes defining and designing the case study. Runeson and Höst (2009) define five major steps when designing a case study research in software engineering:

1. Case study design and planning.
2. Define data collection procedures and protocols for data collection.
3. Collecting evidence.
4. Analysis of collected data.
5. Reporting.

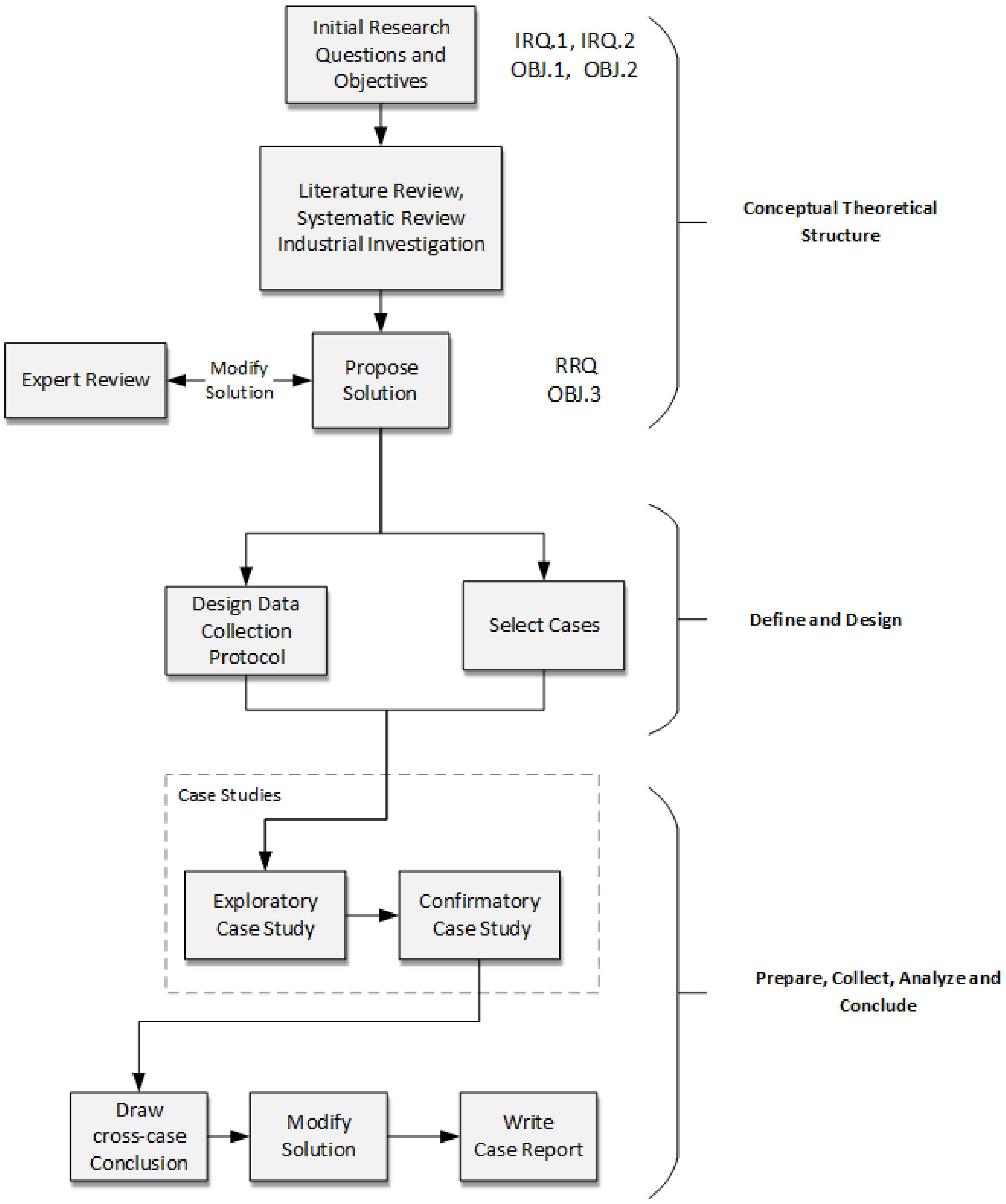

Initial Research Questions and Objectives
IRQ.1, IRQ.2
OBJ.1, OBJ.2
Literature Review, Systematic Review Industrial Investigation
Conceptual Theoretical Structure
Expert Review
Modify Solution
Propose Solution
RRQ
OBJ.3
Design Data Collection Protocol
Select Cases
Define and Design
Case Studies
Exploratory Case Study
Confirmatory Case Study
Prepare, Collect, Analyze and Conclude
Draw cross-case Conclusion
Modify Solution
Write Case Report

**Figure 3.2:** Case Study Research Design (adapted from Yin and Robert (2009)

## 3.2.1 Case Study Design and Planning

Planning the case study includes deciding the data collection methods, an organisation to visit, persons to interview, how interviews should be conducted and unit of analysis. The overall plans for a case study can be formulated in a case study protocol. The case study protocol for this research is shown in Appendix D

The design of the case study must also consider ethical considerations when dealing with confidential information. Runeson and Höst (2009) state that "subjects must explicitly agree to participate in the case study, i.e. give informed consent". In this research, a non-disclosure agreement(NDA) was signed with a case company, and the interviewee from the company signed consent document. Additionally, the research was approved by the School of Informatics and Creative Arts (ICA) Ethics Committee of DkIT.

### 3.2.2 Define data collection procedures and protocols for data collection

The next step in designing the case study includes deciding data collection methods. Having different data sources to choose from, the case study design must explicitly decide the appropriate data collection procedure for the project. Yin (2003) states that "good case studies benefit from having multiple sources of evidence." The findings from different sources need to be rechecked. This is called "triangulating or establishing converging lines of evidence" Yin (2003). In software engineering case studies, the most applicable data collection methods are interviews, observations, archival data and metrics Yin (2003), Runeson and Höst (2009). The data collection procedures selected for this research are direct observation, interviews and archival records.

Direct observations involve passively recording and documenting the activities that take place during the study. In the exploratory case study and the confirmatory case study, the researcher directly observes the events that have been occurring during the project.

Data collection through interviews involve a series of questions being asked by the researcher based on the topic of interest. Interview questions are formulated from the research questions as *open* and closed questions Runeson and Höst (2009). In this research, semi-structured interviews were conducted to collect case data. With the consents of participants, interviews are recorded and transcribed for analysis.

Archival records data collection often takes a form of computer files, organisational charts and documents from different development phases Yin (2014). Artefacts generated from the case studies are used to analyse the results of the case studies.

### 3.2.3 Collecting Evidence

Data collection can be performed directly or indirectly. Direct data collection method allows the researcher to have direct contact with the subjects and collect data in real time. For the case of indirect data collection method, the researcher collects data without interacting with subjects.

### 3.2.4 Analysis of collected data

The most critical step of performing a case study is analysing the case study data. “The basic objective of the analysis is to derive conclusions from the data, keeping a clear chain of evidence” Runeson and Höst (2009). The analysis is performed on an individual or a group that is being studied by the case based on the unit of analysis. The unit of analysis is a tentative definition that is related to the initial research questions and will be “revisited as a result of discoveries during the data collection” Yin and Robert (2009).

Runeson and Höst (2009) state that to perform analysis, data has to be “coded, which means that parts of the text can be given a code representing a certain theme, area or construct”. The next step includes applying analytic techniques such as pattern matching, explanation building, and time-series analysis. A pattern-matching logic enables us to compare the empirically based pattern of the data that has been collected with the predicted one. Explanation building is used when the case study started with open-ended research questions that would lead to an explanation-building. The time-series analyses involve assembling key events into a chronology and producing array. The array "may not only produce an insightful descriptive pattern but also may hint at possible causal relationships".

During the process of analysis, a small set of generalisations can be formulated, eventually resulting in a formalised body of knowledge. Generalisations require carefully constructed claims, and the "claims must be presented soundly and resist logical challenge" Yin (2003). This research is formulated on the bases of open-ended research questions. As a result, we will use an explanation building technique to analyse the collected data.

### 3.2.5 Reporting

The reporting of the case study "should include sufficient data and examples to allow the reader to understand the chain of evidence" Runeson and Höst (2009). The chain of evidence should result in drawing case conclusion and a possible update to the proposed solution. The result of the proposed case studies will be analysed, and generalisations and claims will be reported based on the chain of evidence.

### 3.2.6 Summary of Selected Strategy

This chapter presents the selected research strategy. The research questions and objectives drive the initial theoretical and conceptual structure. Following the definition of research questions and objectives, a systematic review and industrial investigations have been conducted to answer the initial research questions, IRQ.1 and IRQ.2. In order to address the RRQ, the research proposed a process using a combination of suitable agile practices. The process is evaluated through expert review and case study.

Table 3.1 summarises the steps of the case study research and the research instruments that will be used. The next chapter will present the proposed process.

| **Case Study Step** | **Selected Research Instrument** |
|---|---|
| Case Study Design and Planning | The case study protocol, as shown in Appendix D. |
| Data collection procedures | Semi-structured interviews and archival records. |
| Collecting Evidence | Direct and indirect data collection. |
| Analysis of collected data | Explanation building technique. |

**Table 3.1:** Summary of Selected Research Instruments

# Chapter 4

# Proposed Solution

## Introduction

This research initially investigates the challenges of embedded safety critical software development through the SR to address IRQ.1 and IRQ.2. From the challenges identified, the research focuses on addressing multiple stakeholder communication of embedded medical software development which is defined in the RRQ as:

RRQ : How can we support multiple stakeholder communication during requirements analysis of embedded medical software development using a combination of suitable agile practices?

In order to address the RRQ, the research proposed a process using a combination of suitable agile practices. On the following section, the high-level descriptions of the proposed process will be presented.

## 4.1 High-Level View of the Proposed Process

This research is proposing a process, called Sync-Up process, to improve multiple stakeholder communication of embedded medical software development. At the high level, the process is composed of three major phases, PHASE 1, PHASE 2 and PHASE 3. Figure 4.1 shows a high-level view of the Sync-Up process.

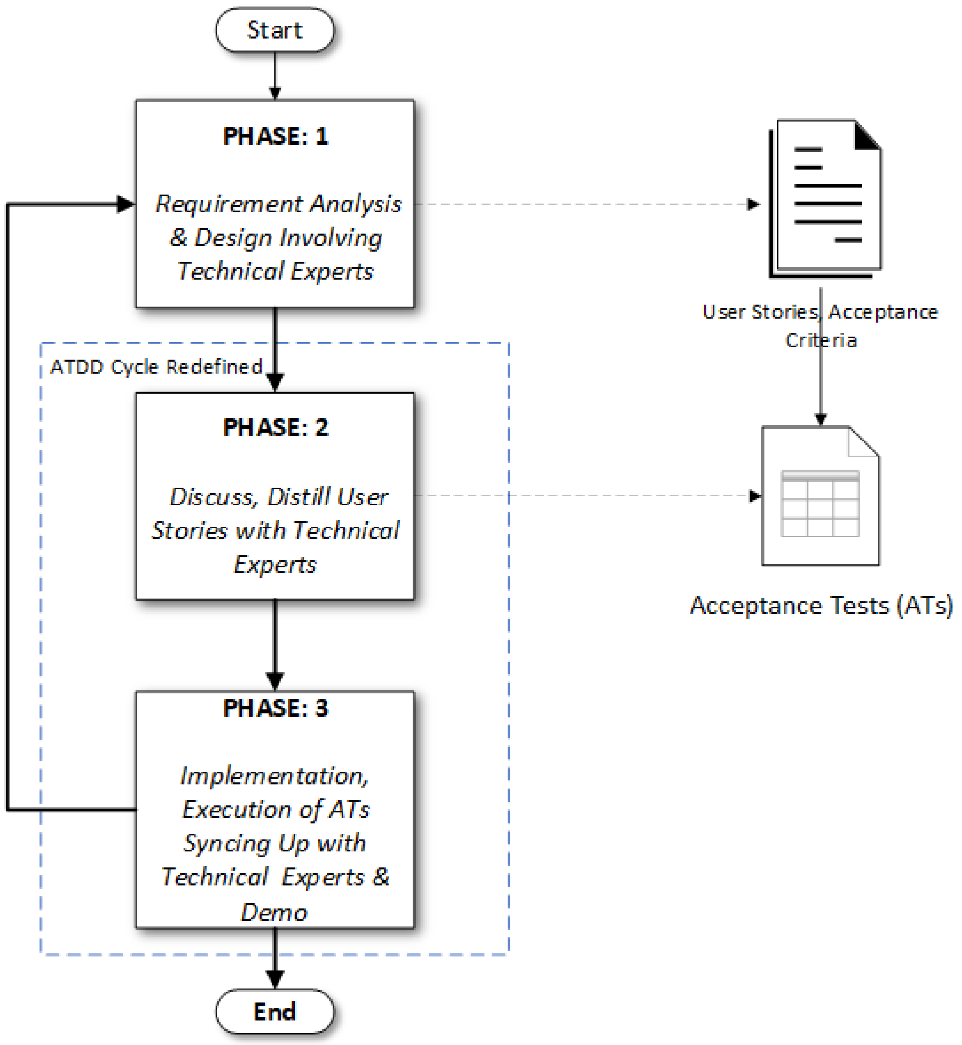

Start
PHASE: 1
Requirement Analysis & Design Involving Technical Experts
User Stories, Acceptance Criteria
ATDD Cycle Redefined
PHASE: 2
Discuss, Distill User Stories with Technical Experts
Acceptance Tests (ATs)
PHASE: 3
Implementation, Execution of ATs Syncing Up with Technical Experts & Demo
End

**Figure 4.1:** High-Level View of Sync-Up Process

PHASE: 1 involves the initial requirement analysis and design stage. During this stage, the embedded cross-functional team composed of software developers, hardware and firmware engineers and product team will collaborate and analyse requirements. The inclusion of technical experts will help the team to consider the *technical feasibility* of the requirements, such as *what works* and what *doesn't work* from the perspectives of different technical members of the team. Additionally, involving technical stakeholders during the early stages will help in reducing the possibility of rework during the later stages of the development.

PHASE 2 and PHASE 3 are defined on the foundation of ATDD. In standard ATDD, originally defined by Hendrickson (2008), the developer, customer and tester will gather to discuss and distill on requirements such as user stories and elaborate *examples* and *acceptance tests (ATs)* of the user stories Pugh (2010), Gärtner (2012). Additionally, they will also collaborate on the development and demo stages. A typical embedded project, on the other hand, has additional roles such as hardware and firmware engineers in addition to the standard ATDD roles. In these phases, the standard ATDD steps are redefined to involve experts such as the hardware and firmware engineers during the discuss and distill stages of the ATDD cycle.

For example, in PHASE 2, the technical experts collaboratively discuss and distill on the user stories from a *technical feasibility* point of view and elaborate the examples and ATs. The discuss and distill activities with the technical experts will help the team to consider detailed technical specifications and write the examples and ATs. The ATs will be elaborated further with additional inputs from the technical experts. This will make the ATs acceptable by all members of the embedded team and provides a channel for communication between the diverse members. Additionally, the ATs will keep the developers on the right track during later stages of development.

In PHASE 3, the ATs will be implemented. Additionally, each member of the embedded cross-functional team will sync up on different implementation issues and dealing with bugs until all ATs are passed. Once all tests are passed, the demo stage helps in ensuring that the feature has been built according to the previously set requirements and expectations.

The detailed description of the Sync-Up process includes a detailed definition of the three phases using a combination of agile practices that are identified through the SR shown in Table 2.7. The next section presents a detailed description of the proposed process.

### 4.1.1 Detailed Steps of the Proposed Process

The detailed steps of the process are defined using agile practices identified from the SR. PHASE 1 is defined further using agile practices that are reported to be used for requirement analysis and design. PHASE 2, on the other hand, is defined using agile practices that are reported to be useful for enabling collaboration between multiple stakeholders with diverse knowledge. Figure 4.2 shows the detailed steps of the proposed process. The next subsection presents a description of the steps.

#### PHASE 1: Requirement Analysis and Design

##### Step 1.1: Define User Stories

In Step 1.1, user stories will be written for the feature that is under consideration. When writing user stories, having a combination of technical experts from each domain will help the team write the user stories with a complete understanding of the system from the embedded, hardware and software perspectives. The user stories will be written using appropriate formats discussed in subsection 2.5.7.

##### Step 1.2: Prioritise, Estimate User Stories

After defining the user stories, the team will prioritise the user stories based on features with more business values and dependencies. The team will also estimate user stories. Estimation is a process of deciding the amount of effort it takes to complete the implementation of the user stories. The embedded, hardware or product members will

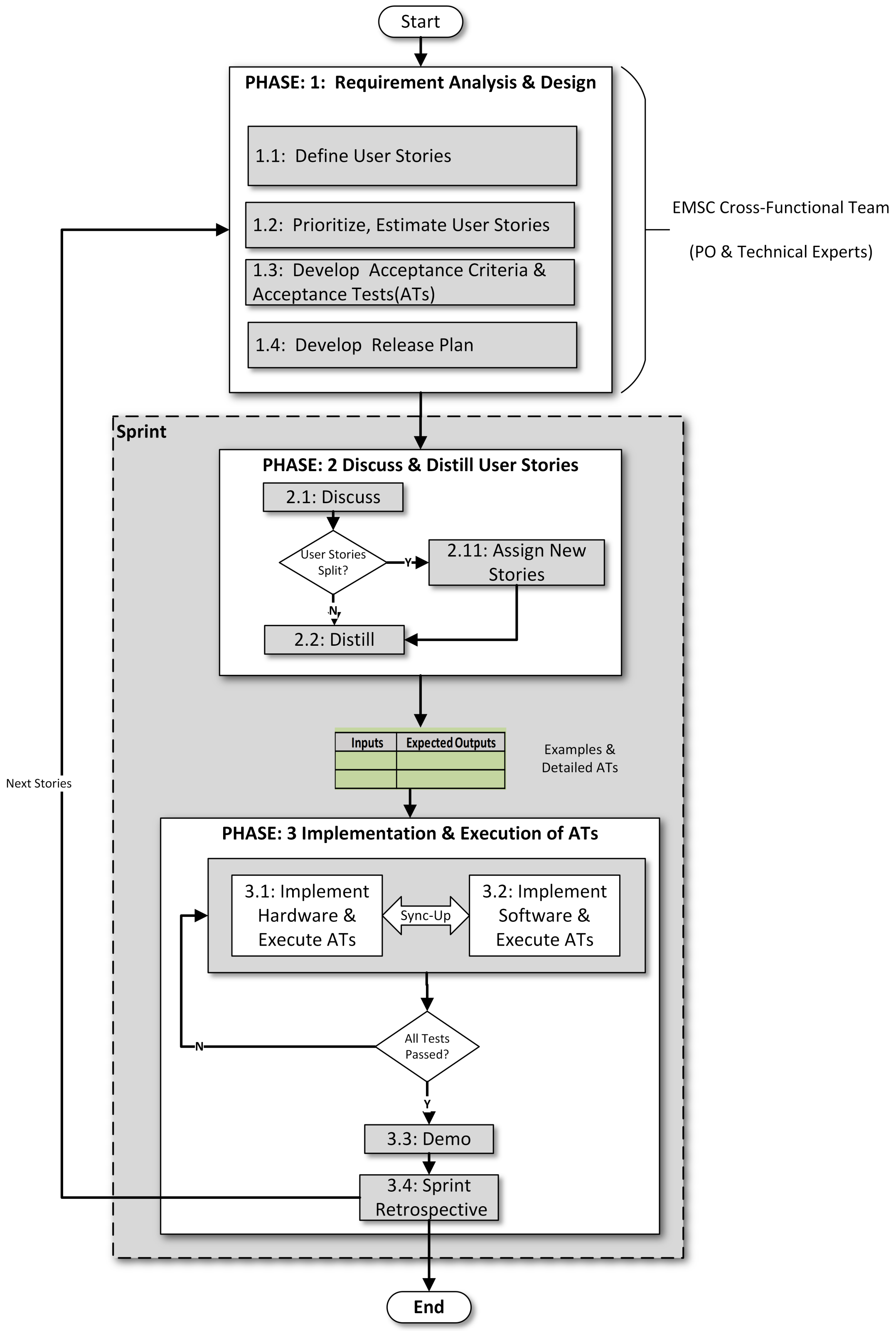

Start
PHASE: 1: Requirement Analysis & Design
1.1: Define User Stories
1.2: Prioritize, Estimate User Stories
1.3: Develop Acceptance Criteria & Acceptance Tests(ATs)
1.4: Develop Release Plan
EMSC Cross-Functional Team
(PO & Technical Experts)
Sprint
PHASE: 2 Discuss & Distill User Stories
2.1: Discuss
User Stories Split?
Y
2.11: Assign New Stories
N
2.2: Distill
Inputs
Expected Outputs
Examples & Detailed ATs
Next Stories
PHASE: 3 Implementation & Execution of ATs
3.1: Implement Hardware & Execute ATs
Sync-Up
3.2: Implement Software & Execute ATs
All Tests Passed?
N
Y
3.3: Demo
3.4: Sprint Retrospective
End

**Figure 4.2:** Proposed Process

collaborate in clarifying dependencies and help the team to achieve better estimates and priorities.

#### Step 1.3: Develop Acceptance Criteria & Acceptance Tests (ATs)

For the user story, the team will write acceptance criteria. Such acceptance criteria will define conditions that must be satisfied to be accepted by each technical experts. For example, the acceptance criteria that are developed in collaboration with firmware, hardware and product experts will define the scope of the requirements and provide a common set of conditions and reduce ambiguity. The expectations of each technical member will be addressed through the acceptance criteria, and this will help to mark the story as *done*. Additionally, acceptance tests (ATs) will be written for each acceptance criteria. The ATs defines a detailed specification of the acceptance criteria.

#### Step 1.4: Develop Release Plan

The final activity of PHASE 1 is the release plan. The release plan describes which feature will be delivered in the upcoming release. In release planning, a list of user stories being considered in the coming sprints will be decided, and the team will decide to commit on the completion of the selected user stories.

### PHASE 2: Discuss & Distill User Stories

This phase involves the clarification of the user stories and ATs by further discussing before the actual implementation of the user stories. When the sprint starts, each team will implement the ATDD steps, discuss and distill user stories.

#### Step 2.1: Discuss

During the discuss step, the team members will collaborate and discuss with examples and elaborate the ATs. The ATs will help to clarify technical constraints and feasibility from the embedded, hardware and software aspects of the feature under consideration.

Each member of the embedded cross-functional team will contribute to the discussion for a common understanding. The team will elaborate the ATs using agile practice such as cross-functional pairing. As a result of the pairing between the technical experts, the ATs written will have detailed constraints acceptable by the experts.

#### Step 2.11: Assign New Stories

This step is implemented if the previous step (Discuss) result in the splitting of the user stories. The discussion with multiple stakeholders could result in the splitting of the stories and such new stories will need to be assigned to the current or next sprint.

#### Step 2.2: Distill

After defining the examples and elaborating the ATs, members of the team will implement distill step. In this step, the team will capture the examples and acceptance test cases using the notations that work with the test automation framework selected for the project. The two common types of notations are discussed in subsection 2.5.7.

### PHASE 3: Implementation and Execution of ATs

#### Step 3.1 & 3.2 Implement Hardware/Software and Execute ATs

In these steps, members of the embedded cross-functional team will implement the hardware and software components. The implementation also includes the execution of ATs. This stage is executed until all ATs for the current iteration are passed. The implementation of ATs is conducted following the principles of TDD. In TDD, developers first write a failing test. This is followed by writing the minimum amount of code that is required to get the ATs passed (Green). Once the ATs are green, members of the embedded cross-functional team will conduct exploratory testing through cross-functional pairing. For example, the technical expert such as embedded engineer and the software developer can pair to test the behavioural and performance-related parameters from the embedded system perspective.

#### Step 3.3: Demo

Once all tests are passed, the user stories will be marked as done, and the embedded project team will move to the demo step to demo the feature to the product owner (PO) and business stakeholders. This step involves re-validating the ATs and making sure that the required functionality has been delivered.

#### Step 3.4: Sprint Retrospective

The final step of PHASE: 3 involves the implementation of sprint retrospective. It is a continuous improvement activity that is held before the next sprint. The objective of this step is to inspect the previous sprint with regards to people, relationships, process, and tools and plan for improvements on the next sprint. The improvements on the current sprint will be considered on the next requirement analysis and design that is defined in PHASE: 1.

## 4.2 Walk-through of Proposed Process

This subsection will present a walk-through of the proposed process using an ideal embedded system project. The following project defines a smart temperature and humidity control embedded system.

**Smart Temperature and Humidity Control Project**

A company planned to develop a smart temperature and humidity control application (API) that lets users monitor temperature and humidity by remotely connecting with the controller device. For evaluation and testing purpose, the company bought DHT22 temperature and humidity sensor, a microcontroller using Arduino board and a WiFi module for connectivity. The controller has to enable connection for registered users. The data from the sensor will be archived over ThingSpeak[a], an open-source Internet of Things application and API to store and retrieve data using the HTTP protocol over the Internet. Figure 4.3 shows the block diagram of the ideal project. The application, on the other hand, has to enable registered users to read current temperature or humidity, calculation of the average temperature for the day and provide the temperature and humidity history for the week.

[a] https://thingspeak.com/

Figure 4.3 shows the block diagram of the ideal project. The embedded system, shown in the dotted box, is composed of DHT22 temperature and humidity sensor, Arduino microcontroller, WiFi module and power supply. The system will be connected to the ThingSpeak platform that will be accessed by the API. The walk-through of the Sync-Up process for the project will be presented for PHASE:1 and PHASE:2. For PHASE:3, Implementation and Execution of ATs, the walk-through will be presented partially as this phase will require the actual implementation of the project.

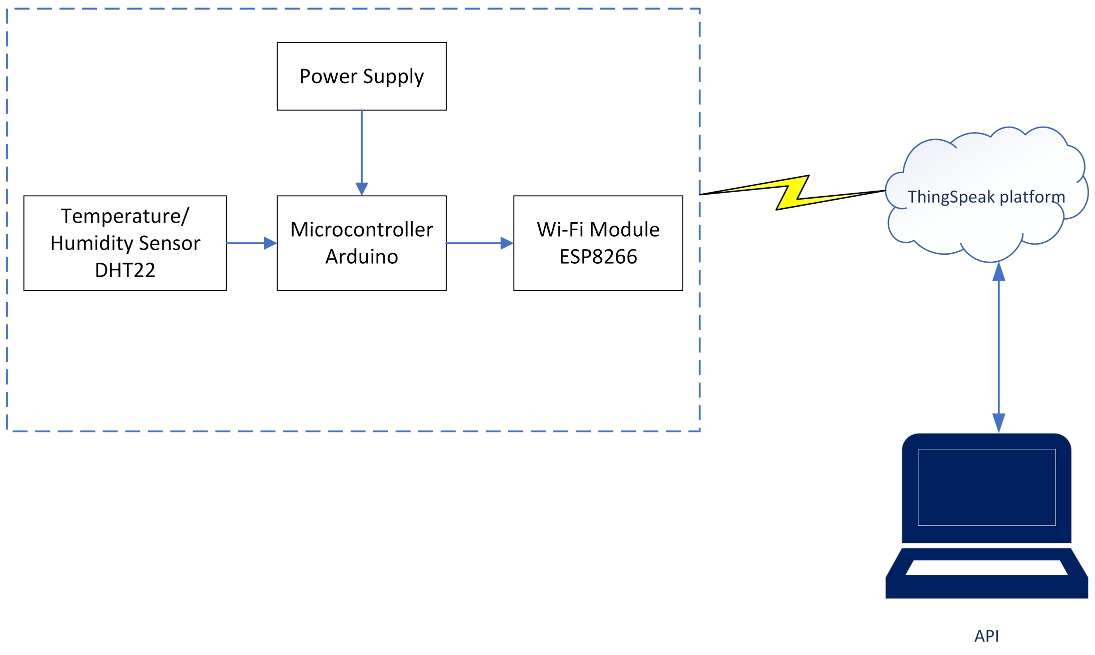
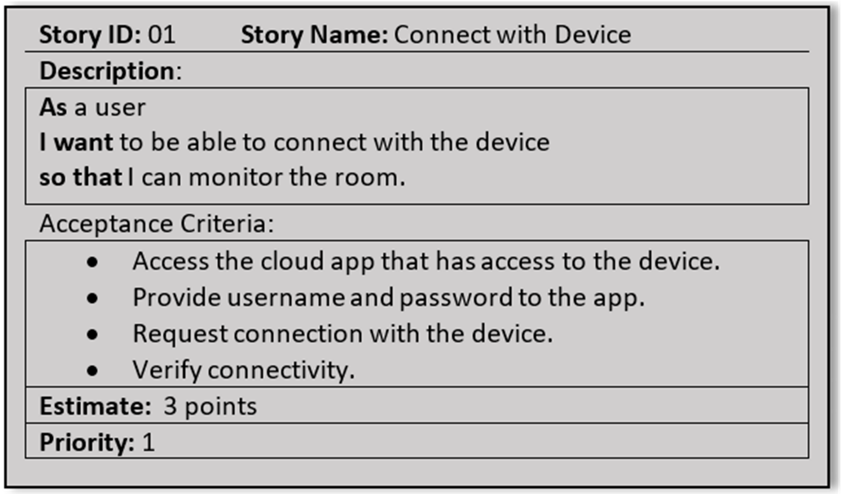


**Figure 4.3:** Smart Temperature and Humidity Control

## PHASE 1: Requirement Analysis and Design

### Step:1.1: Define User Stories

In Step:1.1, the cross-functional team that constitute representatives from product owner (PO), software, hardware, embedded and testing engineers will gather and analyse requirements. The team will write user stories for the features of the project. Using the *Role-Feature-Reason* format of user stories discussed in 2.5.7, the team will define user stories such as: Figure 4.4, Figure 4.5 and Figure 4.6 for some of the features described in the ideal project description. Figure 4.4 defines a user story for connecting with the device. Figure 4.5, on the other hand, defines a user story for reading the current temperature, and Figure 4.6 defines a user story for calculating the average temperature. We can assume that each user stories have been assigned an identification (ID) values of 01, 02 and 03, respectively.



**Story ID:** 01 **Story Name:** Connect with Device

**Description**:

**As** a user
**I want** to be able to connect with the device
**so that** I can monitor the room.

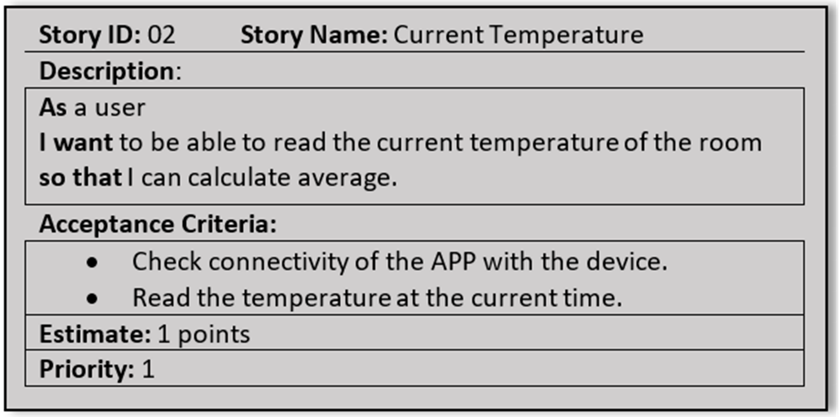
Acceptance Criteria:

- Access the cloud app that has access to the device.
- Provide username and password to the app.
- Request connection with the device.
- Verify connectivity.

**Estimate:** 3 points

**Priority:** 1

**Figure 4.4:** Connect with Device User Story



**Story ID:** 02 **Story Name:** Current Temperature

**Description**:

**As** a user
**I want** to be able to read the current temperature of the room
**so that** I can calculate average.

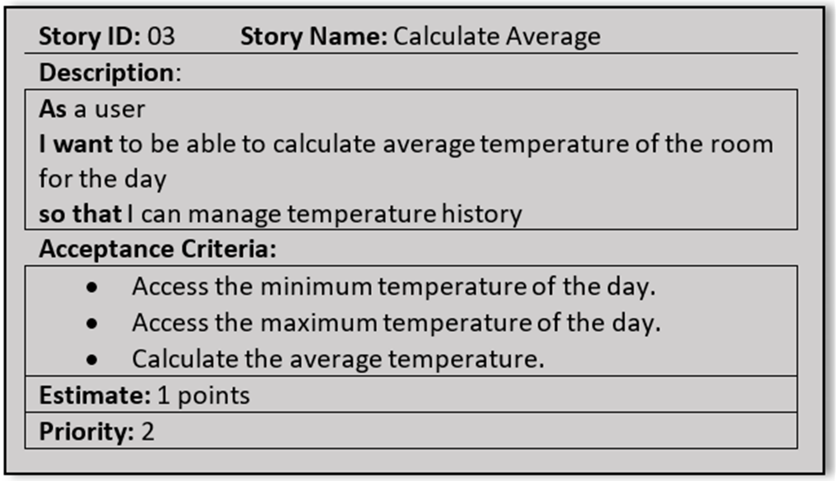
**Acceptance Criteria:**

- Check connectivity of the APP with the device.
- Read the temperature at the current time.

**Estimate:** 1 points

**Priority:** 1

**Figure 4.5:** Read Current Temperature User Story



**Story ID:** 03 **Story Name:** Calculate Average

**Description**:

**As** a user
**I want** to be able to calculate average temperature of the room for the day
**so that** I can manage temperature history

**Acceptance Criteria:**

- Access the minimum temperature of the day.
- Access the maximum temperature of the day.
- Calculate the average temperature.

**Estimate:** 1 points

**Priority:** 2

**Figure 4.6:** Calculate Average Temperature User Story

### Step:1.2: Prioritise, Estimate User Stories

After defining the user stories, the team will perform prioritisation and estimations. The user stories will be estimated with the involvement of all technical members of the embedded cross-functional team. For example, user story 01 defined for *connecting with the device* can be assigned 3 story points as configuring and connecting with the device will need more efforts. On the other hand, the user story 02 defined for *reading the current temperature* and user story 03 defined for *calculating average temperature* can be assigned 1 story points as reading the sensor data and calculating the average temperature can require less effort once the connection with the device have been made. In addition to estimation, the prioritisation of the user stories will also be conducted based on the business need and dependency between different functionalities of the project. For example, user story 02 should be implemented after the user story 01 as we cannot read temperature unless we connect with the device. Additionally, any user story that have a functionality of communicating with the device require the device to be in standby mode and configured properly.

#### Step:1.3: Develop Acceptance Criteria & Acceptance Tests (ATs)

After defining the user stories and performing prioritisation and estimation, acceptance criteria will be written by the cross-functional embedded cross-functional team team composed of customers, product owner, developers, testers and embedded engineers. For user story 01, the following set of acceptance criteria can be defined.

- Access the cloud APP that has access to the device.
- Provide user name and password to the app.
- Request connection with the device.
- Verify connectivity.

The acceptance criteria for user story 02, reading the current temperature, can be used to check if connectivity with the device has been made before trying to read the current temperature.

- Check the connectivity of the APP with the device.
- Read the temperature at the current time.

Once acceptance criteria are defined, ATs will be written using notations that will be selected for the project. From the common notations that are used to write ATs, we will present the walk-through using Fit Tables. Using this notation, ATs are written in the form of tables. An example of AT table for reading temperature from the device at a specific time is shown in Table 4.1.

| **Input** | **Expected Output** |
|---|---|
| *ReadTime* | *Temp()* |
| 0 | Temp Data |
| 1 | Temp Data |
| 2 | Temp Data |
| 3 | Temp Data |
| 4 | Temp Data |

**Table 4.1:** Read Temperature Acceptance Test

Table 4.1 shows the acceptance test case written for user story 02 using the ColumnFixture notation. In this notation, table columns are mapped to fields or methods, and output values will have parentheses. The *Read Time* represents the time, in seconds, at which reading will be performed, and the *Temp* value with parentheses represent the output temperature value that is read at the specified time. On the other hand, Table 4.2 shows the acceptance test case written for the user story 03. The first two columns show the minimum and maximum temperature values and the third column shows the average temperature output.

| **Input** | **Input** | **Expected Output** |
|---|---|---|
| *min* | *max* | *Average()* |
| 25 | 35 | 30 |
| -40 | -20 | -30 |
| 70 | 81 | 75.5 |

**Table 4.2:** Calculate Average AT Case

#### Step 1.4: Develop Release Plan

This step involves the selection of user stories that are going to be implemented during the current sprint. Based on the prioritisation and business needs, the team will select stories that must be considered in the current sprint. For example, we can assume that before working on user stories that implement the application software components of the project, the user stories that define the basic device configuration and communication should be implemented first.

The output of Phase 1 involves a set of user stories that are prioritised and estimated. Additionally, acceptance criteria and ATs, that are defined for the user stories, with release planning that will decide the user stories that are going to be implemented in the current sprint.

### Phase: 2: Discuss & Distill User Stories

#### Step:2.1: Discuss

In this step, the embedded cross-functional team will discuss and distill the user stories and ATs. One of the main benefits of the discuss stage is to perform a detailed clarification on parameters and extreme cases. The involvement of technical experts of the team will help to clarify the ATs of the user stories. For example, for the previous test case written to read the current temperature, the technical experts of the team such as the embedded and hardware engineers can discuss on the sampling period for the DHT22 sensor, which is 2 seconds. This means that for an accurate result, consecutive readings should not be performed in less than 2 seconds. As a result, the team can agree on the delay imposed as a result of the sensor sampling time. The AT shown in 4.1 can be modified to check consecutive readings. Based on the discussion, additional constraint can be added to the initial ATs by adding inputs columns with *CurrentReadTime* and *PreviousReadTime* entries. The early clarification of such additional information, before development starts, will help the team to avoid possible future rework.

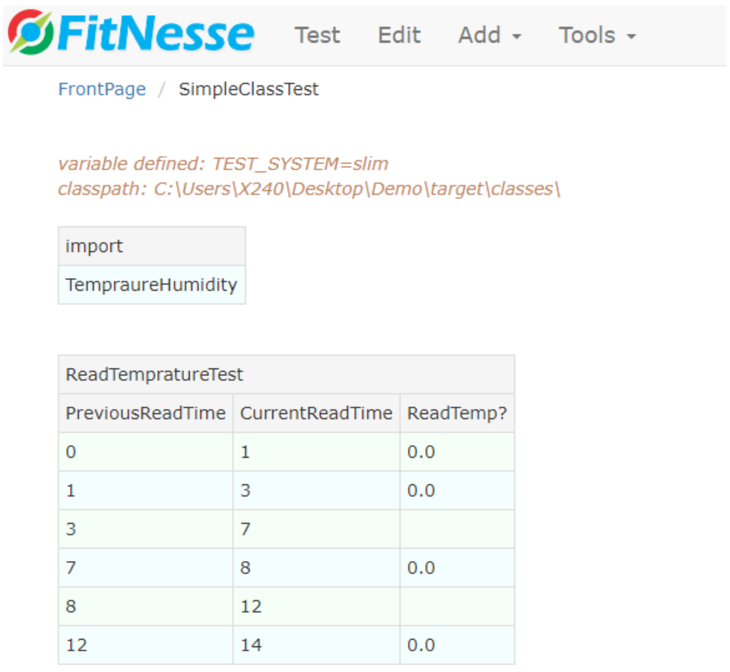
| **Input** | **Input** | **Expected Output** |
|---|---|---|
| *PreviousReadTime* | *CurrentReadTime* | *Temp()* |
| 0 | 1 | 0.0 |
| 1 | 4 | Temp Data |
| 7 | 8 | 0.0 |
| 10 | 20 | Temp Data |

**Table 4.3:** Modified AT Case with New Entries

Table 4.3 shows the modified AT case table. We can observe that from the discussions between technical experts, the initial acceptance test case can be elaborated further with the new information. For example, the expected output values can be assigned empty or 0.0 for readings that are performed in less than 2 seconds.

#### Step:2.2: Distill

This step involves capturing the examples and ATs using the appropriate test automation framework. Some of the common AT frameworks include FitNesse, Cucumber and Robot framework. In this walk-through, the distill step will be presented using FitNesse framework. This framework uses table based AT formats known as Fit Tables. An example of a Fit table for the AT case shown in Table 4.3 is shown in Figure 4.7. Based on the rules defined in this AT case, for reading performed in less than or equal to 2 seconds interval, the system under test (SUT) is supposed to respond with 0.0 temperature value. On the other hand, readings performed in more than 2 seconds should have the current temperature. For the walk-through, we will generate random float values between (0 - 40) to mimic temperature readings.



variable defined: TEST_SYSTEM=slim
classpath: C:\Users\X240\Desktop\Demo\target\classes\

| import |
|---|
| TempraureHumidity |

| ReadTempratureTest | | |
|---|---|---|
| PreviousReadTime | CurrentReadTime | ReadTemp? |
| 0 | 1 | 0.0 |
| 1 | 3 | 0.0 |
| 3 | 7 | |
| 7 | 8 | 0.0 |
| 8 | 12 | |
| 12 | 14 | 0.0 |

**Figure 4.7:** Fit Table for Read Temperature AT Case

## Phase 3: Implementation and Execution of ATs

### Step 3.1 & 3.2 Implement Hardware/Software and Execute ATs

The implementation of hardware and software involves writing the application code and building the embedded components. During the application code development, the ATs will be executed. To execute the ATs using the selected automation framework, an additional code needs to be written for the Fit table in Figure 4.7. Such code is known as a fixture. The path to the fixture class will be defined using the *classpath:* command on the Fit table wiki page. The fixture code will then test the SUT for all test scenarios.

The SUT code that is developed for this walk-through is an abstract application that mimics the ideal smart temperature and humidity control application. During the implementation of the application, the AT cases needs to be executed until all tests cases are passed. A passing and failing test cases are shown in Figure 4.8 and Figure 4.9. In Figure 4.8, the AT case is passed as the difference between current read time and previous read times are more than two seconds, and the SUT responded with the temperature data as expected.

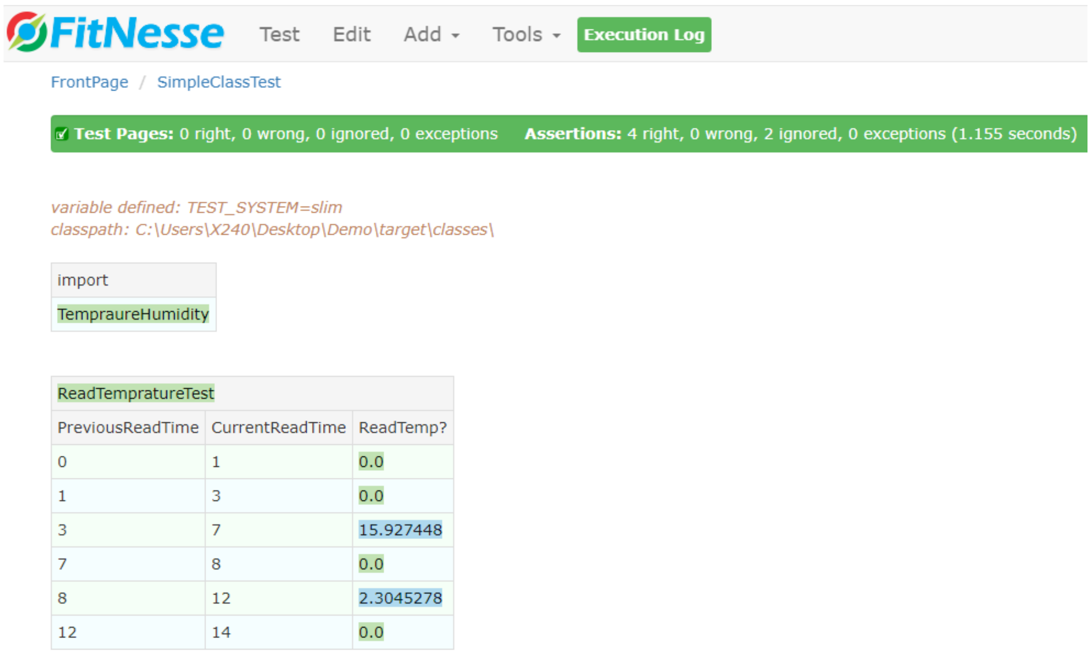
During the implementation phase of both software and embedded components, the SUT could respond differently than the previously agreed acceptance criteria. Let us assume the temperature sensor is sending irregular temperature data every one second due to temperature sensor sampling error (hardware problem) or application software error.

The previously agreed AT case should generate an error, as shown in Figure 4.9. For example, the first row shows temperature reading performed in one-second duration. The expected result, (0.0) and actual result (26.69) are different, and the test fails. On the initially developed AT case, the teams have agreed to have common acceptance criteria and AT that should be fulfilled to mark the user story as complete. The elaborated AT cases from the discussions with the experts clarified that readings should be performed in more than two-second duration for correct sampling. The AT cases will help the team to track down the issues based on initially agreed acceptance criteria.

Once all acceptance tests are passed for the selected user stories in the current sprint, the final step of Phase 3, Sprint Retrospective, will be implemented. The team will analyse the current sprint for possible improvement and will go back phase 1 for the next set of user stories.

### 4.2.1 Summary of the Walk-through

In this subsection, a walk-through of the Sync-Up process was presented for an ideal embedded system project. The walk-through was also demonstrated using an automated acceptance testing framework known as FitNesse. The expected artefacts of each step have been discussed. One of the main objectives of the proposed process is to support the collaboration of multiple stakeholders that are involved in an embedded medical project. Through the steps of the process that allows cross-functional embedded teams to collaborate and sync up, the walk-through has shown that the acceptance criteria and



FrontPage / SimpleClassTest

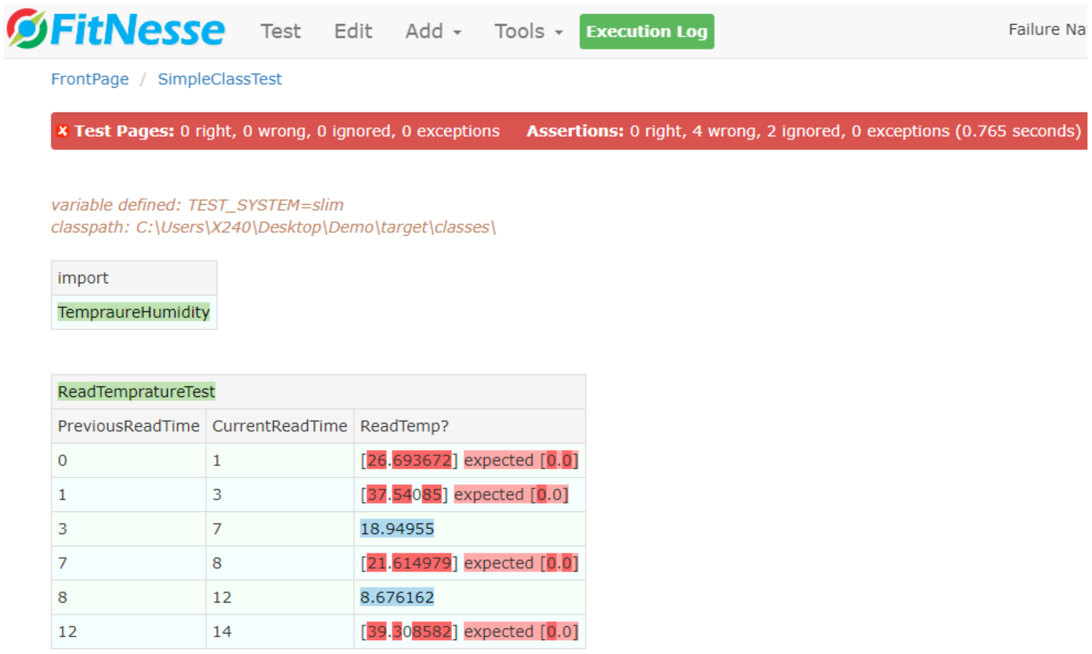
**Test Pages:** 0 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 4 right, 0 wrong, 2 ignored, 0 exceptions (1.155 seconds)

*variable defined: TEST_SYSTEM=slim*
*classpath: C:\Users\X240\Desktop\Demo\target\classes\*

| import |
|---|
| TempraureHumidity |

| ReadTempratureTest | | |
|---|---|---|
| PreviousReadTime | CurrentReadTime | ReadTemp? |
| 0 | 1 | 0.0 |
| 1 | 3 | 0.0 |
| 3 | 7 | 15.927448 |
| 7 | 8 | 0.0 |
| 8 | 12 | 2.3045278 |
| 12 | 14 | 0.0 |

**Figure 4.8:** Passing Test Case



FrontPage / SimpleClassTest

**Test Pages:** 0 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 0 right, 4 wrong, 2 ignored, 0 exceptions (0.765 seconds)

*variable defined: TEST_SYSTEM=slim*
*classpath: C:\Users\X240\Desktop\Demo\target\classes\*

| import |
|---|
| TempraureHumidity |

| ReadTempratureTest | | |
|---|---|---|
| PreviousReadTime | CurrentReadTime | ReadTemp? |
| 0 | 1 | [26.693672] expected [0.0] |
| 1 | 3 | [37.54085] expected [0.0] |
| 3 | 7 | 18.94955 |
| 7 | 8 | [21.614979] expected [0.0] |
| 8 | 12 | 8.676162 |
| 12 | 14 | [39.308582] expected [0.0] |

**Figure 4.9:** Failing Test Case

AT cases developed will help the involved teams to have a common language to drive the overall development process.

## 4.3 Conclusion of Proposed Solution

This chapter presents the description of the proposed process that is developed to address the research question, RRQ.

The chapter began by describing the high-level and detailed description of the process. Additionally, a walk-through of the process was presented for an ideal embedded system project. The walk-through shows the expected artifacts and expected benefits of the proposed process. The next chapter will present the first part of the validation process using expert review.

# Chapter 5

# Expert Review

## Introduction

This chapter presents the validation of the proposed process through expert review. The expert review will help in evaluating the proposed process by leading experts that have experience in agile and embedded software development. The review process includes the selection of experts, presenting the walk-through of the proposed process to the experts and addressing the improvements suggested by the experts. The next subsection presents the expert selection process. This is followed by the review process, and finally, the improvements suggested by the experts will be presented.

## 5.1 Expert Selection

The search for the experts was conducted through LinkedIn[1], a social network for professionals. The researcher contacted candidate experts that satisfied the criteria defined in the following subsection.

[1]https://www.linkedin.com

### Criteria for Selecting Experts

The experts have been selected based on their knowledge and experience on agile software development within embedded safety critical domains. The main criteria that have been used for selecting the experts were:

- Having many years of experience implementing agile software development either as a product owner, scrum master, developer or consultant.
- Having experience in embedded safety critical domains.
- Having experience in implementing or coaching the implementation of specific agile practices such as ATDD.

The expert identification, communication, and review was conducted over the period of ten months. Connection requests were sent to 17 experts. Additionally, the researcher advertised the expert review request on closed groups of embedded systems. Initially seven experts agreed to be part of the expert review. During the later stage of the communication, four of the experts agreed to review the proposed process.

Once experts were identified, they were sent an expert consent form for their participation and willingness. The expert consent form that was sent to the experts is shown in Appendix E. Four leading experts have been involved in reviewing the Sync-Up process. A short synopsis of each expert profile will be presented as follow.

### Expert 1

Expert 1 has over twenty years of experience in different industries where he worked in many different roles including, developer, tester, analyst, product manager, test manager and agile/lean coach. The expert had a well-known case study on the implementation of ATDD.

### Expert 2

Expert 2 has fourteen years of experience in software testing of embedded medical devices such as injection devices and infusion pumps, smart lighting embedded software and various embedded test automation tools projects. At the time of the review, the expert was working on two projects. On one of the projects, he was a scrum master that includes eight people composed of software developers, software testing and product owner. On the other project, he was working as a delivery manager which involves hardware engineers, firmware developers, mobile app developers and quality assurance.

### Expert 3

Expert 3 is a speaker, consultant and author of a dozen books on agile, Lean and managing high-technology product development. The expert has helped managers, teams and companies to move to an agile approach by applying a pragmatic approach. She has helped companies in embedded safety critical domains.

### Expert 4

Expert 4 has twenty years of experience in software and embedded system design for instrumentation in safety-critical industrial and medical applications. She is among the first to apply agile methods to embedded systems development. She has led agile change initiatives beyond software development in safety-critical systems such as avionics, factory automation, medical & defence system, and teaches modern agile approaches like mob programming, agile hardware, and Lean development methods.

## 5.2 Review Process

For Expert 1, one meeting was conducted, and the researcher presented the walk-through. The expert was then asked to provide his advice and suggestions. For Expert 2, Expert 3 and Expert 4, two meetings were conducted using video conferencing tools Skype and Zoom. The first meeting was an introductory session where the researcher and the experts would get to know each other. In this meeting, the experts were asked questions related to their experience in agile software development, embedded software development and challenges faced by the experts. On the second meeting, the researcher presented a walk-through of the proposed process, and the experts were asked to provide their advice on the proposed process. The experts were asked to point out the deficiency observed, benefits, improvement and other suggestions they would like to add. The list of presented questions is given in Appendix F. With the consent of the experts, the video conferences were recorded and transcribed for reference.

The summary of the review conducted with each expert is presented in the following subsections.

#### Expert 1

This expert has extensive experience in the implementation of ATDD and has been helping companies to improve their collaboration by implementing ATDD practices such as discuss and distill. With the limited availability, the expert was only presented the walk-through of the proposed process.

When asked to comment on the benefits of the Sync-Up process, the expert stated that he believes the proposed process can work and help the embedded cross-functional team to deliver faster. The expert went on to state that the main reason we pair people is because “we want to inject quality” at the beginning rather than detecting problems at the end. He stated that during the development phase, if we “find issues late”, there will be a lot of rework. By injecting some quality at the beginning of this phase as a result of pairing, we can remove possible rework. The expert noted that the impression he got from leaders is, if people are paired, then they will only get one thing done instead of two things that they could work and “this could be a problem”. To overcome this, the expert emphasised the importance of “coaching managers” and explaining the concepts from Lean on limiting work in progress (WIP). He stated that by limiting WIP, we can actually get more things done. According to the expert, occupying the members of the cross-functional team only for about 70% of their time is more likely to produce more work than if we occupy them for 100%.

For the remaining experts, two interviews were conducted. The first interview was conducted to gather the challenges and experience faced by the experts. On the other hand, the second interview was conducted for the comments and suggestion on the proposed process. The result of the first interview conducted with the experts will be presented in the following subsection.

### 5.2.1 Challenges and Experience

#### Expert 2

The expert stated that he has been working with diverse teams such as hardware engineers, firmware developers, application software engineers and scientist teams. The expert noted that from his experience hardware teams were not following agile and they were working at “their own pace”. The firmware teams, on the other hand, would work based on the initially available hardware. Additionally, the application software teams were completely relying on the embedded software. The expert stated that such diverse stakeholders were not integrating their tasks properly.

The expert gave an example from his previous project experience. He stated that on the project to develop the smallest endoscope, the teams initially agreed to use an

Android operating system for displaying the real-time image of the endoscope camera. The selection of this operating system would have “some delays in a microsecond” on the processing of the image. During the requirement analysis and development phases, the stakeholders had all agreed on the delay and the team delivered first build and went for a trial with doctors and scientists. Once it went into the doctors and scientists “they found that this delay was unacceptable”. The expert stated that this *change* cost the team around four months of *delay* because they needed to change the operating system from Android to Linux completely. The change in the operating system required additional changes to the video connection and communication protocols. The expert stressed that the involvement of some of the stakeholders, such as the scientist team, at the later stage created “huge delay” and a clear example of the communication gap and miss-collaboration of diverse members.

When asked to comment on the benefits of the syncing up, the expert stated that most of the time failure occurs because stakeholders were not reviewing and analysing requirements and AT cases. The expert referred to another project where he was involved. In this project, the teams initially agreed on AT cases and developers started based on this test. The expert stated that when about 40% of the development was completed, a reviewer team from stakeholders started reviewing the AT cases and found that the AT test cases were “not something which they were expecting”. The expert went on to state that this change has ended up *wasting* about 40% of the time of the software developers, firmware engineers and test protocol designers because the detailed test plan development was already started. The expert stated that all members of the stakeholders should sync up early and agree on the user stories and AT cases before development is started, to avoid the cost of rework.

**Expert 3**

This expert was initially asked about the experience of implementing agile and the challenge of multiple stakeholder communication. The expert responded that from her experience working with different embedded clients, hardware teams have always been "almost separated from firmware teams". The software team, on the other hand, would have often been in a third location. The expert also stated that even if the teams were in the same location, they would have been on "different forces". The separation of such diverse teams was creating a number of long integration groups. The expert stated that for embedded software development, most problems show up after the product is in the field, and this would make it "really difficult to tell where the problem arises".

When asked to comment on the benefits of the syncing up, the expert stated that to have a coherent user story that has important components from the architecture, we need to sync up various layers such as the application program interface (API), middle-ware and the platform which encompasses both firmware and hardware. The expert stated that user stories should impact on the architecture, not just the software side. Additionally, the expert noted that ATs should also include the hardware acceptance, the firmware and the mechanical aspects of the system.

The expert used an example of a client that implemented user stories for software and firmware and design by contact for the hardware. The client was using separate Kanban for software and firmware user stories, so the firmware was always verified in advance of the software. The expert advised the client that the firmware and software teams have to work together as a cross-functional team, otherwise, they would run into problems because the cycle time of the firmware Kanban and the software Kanban will be different. The expert went on to state that the client didn't like her advice because the client was not looking at the user story cycle time, merely the software and firmware. The expert stated that "we might end up with big teams involving software, firmware and hardware developers, but all multiple stakeholders should come together as a cross-functional team and create the user stories". Additionally, having the firmware and mechanical experts will help the team to consider the implications of the firmware and mechanical components on the user stories.

**Expert 4**

The expert stated that she worked with diverse members such as software admins, software engineers, firmware engineers, electronics designers, mechanical engineers, materials scientists and mathematicians. She was involved in projects covering different stages of the software development life-cycle (SDLC) such as requirement analysis, design, and implementation.

The expert gave an example project on multiple stakeholder communication. She stated that in the project the system engineer wanted all groups to do their design for a scientific instrument that needs to produce each measurement within tight performance time constraint (a few seconds). The expert went on to state that the system engineer was not willing to tell other teams how much of that time could be allocated to the software they were designing. There were "too many unknowns" for him to do this, and no other disciplines outside software could work incrementally. The expert stated that this created a big strain in the project team because this situation could only be addressed by a cross-functional team where all disciplines could cooperate in a fluid way to "emerge" a workable solution in an iterative way.

When asked to comment on the benefits of the syncing up, the expert stated that from her experience, the most expensive problems boil down to "stakeholders having misaligned views or goals". In her experience, it was a rampant problem, and the Sync-Up process can help to point out this challenge. Additionally, having worked in flight simulation that includes mathematical modelling, design and various engineering types, the expert stated that such disciplines that go into simulation work can also benefit from the Sync-Up process.

Table 5.1 summarises the project experiences of the experts regarding multiple stakeholder communication.

| **Factors of Multiple Stakeholder communication challenge** | **Effects on Projects** |
|---|---|
| Involvement of stakeholder at later stage of development. | Delay on project timeline, wasting developers time. |
| Separation of hardware, software and firmware teams (Separate Kanban). | Long integration groups. |
| Too many unknowns between between different disciplines. | Strain on project timeline. |
| Stakeholders having misaligned views or goal. | Extra effort required when problem arise. |

**Table 5.1:** Factors of Communication Challenges and Effects on the Projects

### 5.2.2 Comments on a Process

In the first part of the interview conducted with each expert, we have explored the experience of the experts on multiple stakeholder communication. The second section of the interview includes the researcher presenting the walk-through of the proposed process, and the experts were asked to give their comments on the proposed process. The summary of the comments given by each expert will be presented in the following subsections.

#### Expert 1

Being an expert on implementing ATDD, this expert was asked to give his advice on the steps that involve ATDD. In his experience, he preferred to do the examples with the discuss and distill steps, after starting the sprint. The expert noted that the discuss and distill practices are very demanding activities, and we cannot expect people to implement them for more than an hour or more. Based upon his experience, he found that it's "more effective" to discuss and distill based on the user story bases. The expert went on to state that when we start our sprint, we can take a user story and get all the stakeholders together and design the examples. Once each stakeholder is happy with the examples, developers will start writing the functional tests cases and implement the code for those specific user stories.

The expert also suggested that the discuss and distill activities could result in the splitting of the user stories. The expert went on to state that when we define user stories and get people to write examples, we will find out very soon whether a user story is too big and needs to be split. According to the expert, if there are three or more examples in a single user story then the user story is “already too big”. The expert suggested that the cross-functional team can identify a set of examples, and by looking at the examples, the team can find a way of splitting the user stories in two or three. The new user stories can be added to the current or the next sprint depending on how priorities are stored.

**Expert 2**

The expert stated that Phase 1 can be implemented in two ways. The first way suggested was that members of stakeholders all come together and discuss and create their requirements. The expert went on to state that the best scenario would be if the product owner (PO), Scrum Master, a technical expert from each domain like the hardware lead, the firmware lead, application lead and testing member are all involved and sync up to create user stories and ATs. The expert went on to state that hardware teams usually work in multiple activities and depends on tools, labs and also their work usually depends on vendors. The expert stated that to incorporate the changes in line with members such as firmware and software teams, syncing up is important.

The second way suggested by the expert was if multiple stakeholders create a cross-functional team and write user stories and ATs. From his experience, the expert noted that creating a cross-functional team in the embedded domain is difficult. The expert went on to state that the hardware team’s dynamics are different than that of the software team and their working pattern is also different. According to the expert, the hardware team generally work in a pure waterfall model. The software team, on the other hand, follow the design output of the hardware team and adopt the changes proposed.

Despite the difficulty of creating a cross-functional team, the expert suggested that the activities in Step 1, which are defining user stories and ATs, can be implemented using cross-functional teams composed of each technical expert.

### Expert 3

A walk-through of the proposed process was presented to the expert, and the expert was asked to provide the deficiency and recommendations observed. She stated that if hardware, software and firmware teams in their silos, separately sync up their activities after some period of time, “that’s not agile”. The expert suggested that the activities in PHASE:1 should include all members of stakeholders in one cross-functional team. The expert stated that in this phase syncing up is “non-existent” as we’re part of a cross-functional team with a representative of all layers of architecture. The expert also noted that the discussions on the user stories such as the discuss and distill stages in PHASE:2, should also include the entire team.

For PHASE:3, the expert stated that we might need “little sync ups” frequently. In her experience, hardware and firmware teams iterate on the design and create tooling for simulations. The expert stated that as long as simulations are available, hardware and firmware teams can have something to share with other members such as software developers and syncing up will be possible.

### Expert 4

The expert stated that in practice, some of the sequencing of steps often cannot be done as planned. For instance, breaking user stories into smaller ones will cause re-estimation, and this itself will raise questions that cause a rethink of the requirement that preceded the first user story definition.

The expert went on to state that engineering is a creative endeavour, but is not usually seen as such. For example, in a TV series, the actors start to mould the characters, causing the writers to get new insights, and so on to the point that in the best instances no one can say how many cycles through the write-edit-act loop have been done. Similarly, the agile embedded software teams end up making small utilities that help to allow hardware and software to proceed independently via strong interfaces with frequent cycles of the hardware test. This will allows better hardware and software ideas to feed to the next set of user stories. The expert stated that the Sync-Up process definition has to allow an indeterminate number of cycles around the creative loop.

**Summary of Comments**

The comments stated by the each experts is summarised as as follow:

- The implementation of the discuss and distill steps should be conducted after starting the sprint.
- The embedded teams should discuss and distill on the user story bases.
- The discuss and distill activities could result in the splitting of the user stories that can be added to the current or the next sprint.
- Phase 1 can be implemented in two ways. In both ways, creating a cross-functional team is difficult in the embedded domain.
- The activities in Step 1, can be implemented using cross-functional teams composed of each technical expert.
- For PHASE:3, we might need “little sync ups” frequently as long as simulations are available.
- Some of the sequencing of steps often cannot be done as planned but embedded software teams have to make small utilities that allow hardware and software to proceed independently.

## 5.3 Improvement

All the four experts were presented with the walk-through of the initial proposed process which is shown in Figure 5.1. The experts were then asked to give their advice on any improvements they may suggest. The improvements and suggestions made by each expert were taken into consideration and applied to the initial version of the proposed process. The improvements made after the suggestion of the experts will be presented in the following subsections.

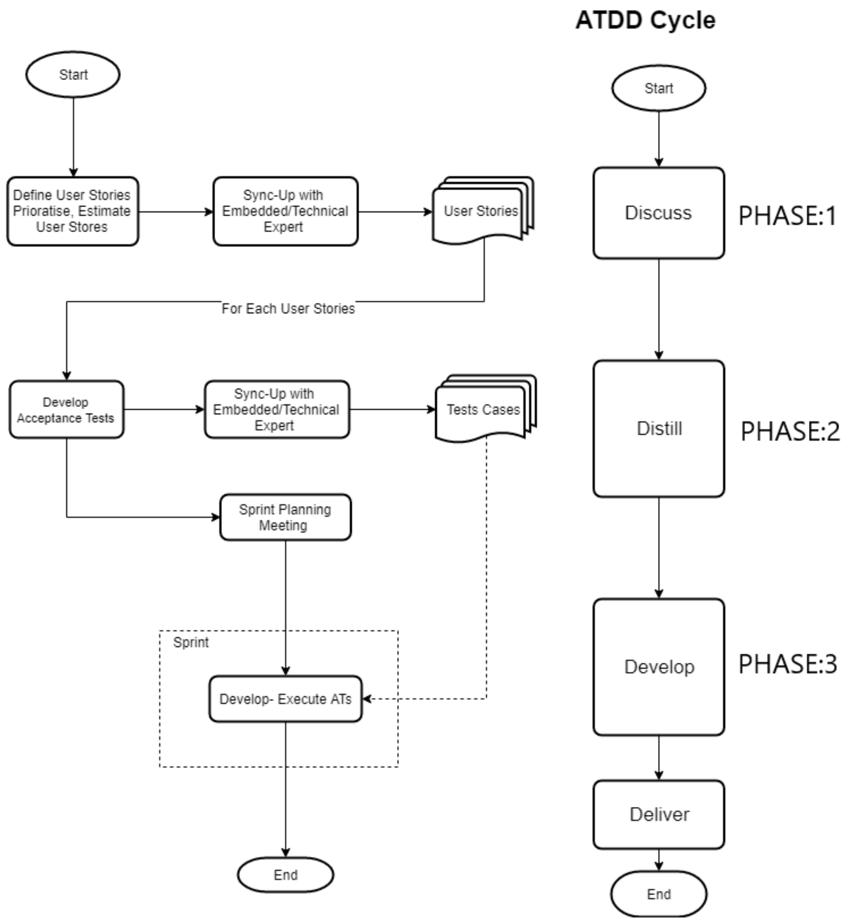


**Figure 5.1:** Initial Version of Proposed Process

### 5.3.1 Improvement 1

Expert 2 stated that the activities in PHASE:1 can be implemented using a cross-functional team. This expert noted the difficulty of creating a cross-functional teams in embedded system projects. The expert suggested that he prefers if the hardware lead, firmware lead, application lead and the product owner all come together and create the user story and ATs. On the other hand, the expert highlighted the importance of syncing up during the development phase.

According to Expert 3, the way diverse teams sync up in different stages has to be clarified. She suggested that the initial steps of defining the user story and ATs should be conducted using a cross-functional team. On the other hand, the expert stated that during the development phase, PHASE:3, we will need little sync ups as separate silos with simulations on the hardware and continuous integration with the software. Additionally, the other experts also stated that multiple stakeholders need to sync up during development frequently in their separate silos. The experts stated that the frequency of syncing up during the development phase will depend on the availability of simulations and cost of prototyping. The experts stated that when the hardware is not ready, simulation and prototyping of the hardware can be used by software and firmware teams in advance and development can be started. As highlighted by Expert 1, for a complex machine, prototyping can be very expensive. On the other hand, for small devices such as wearable items, prototyping can be done using evaluation boards.

### Update on Process

On the initial version of the proposed process, the Sync-Up process was designed to only involve the embedded engineer to sync up with the development teams in all three phases. The changes to the initial version of the proposed process made as part of the feedback from the experts, attempted to resolve this issue, i.e. a cross-functional team will be formed and the team will implement PHASE:1 and PHASE:2 using cross-functional pairing.

### 5.3.2 Improvement 2

All experts stated that the PHASE:2 could lead to the splitting of the user stories and such new user stories need to be assigned to the current or the next iteration. Additionally, the experts stated that it's better to conduct the discussion of the user stories when the Sprint starts.

#### Update on Process

On the initial version of the proposed process, the splitting of user stories was not considered. On the latest version of the proposed process, an additional decision was placed after the discuss step to check if the new user stories are being added to the current or next sprint. On the other hand, the discussion of the user stories is modified to be conducted when the sprint starts. The latest version of the proposed process is shown in Figure 4.2.

## 5.4 Conclusion of Expert Review

The validation of the Sync-Up process was conducted through expert reviews. The experts involved have shared their experiences on the challenges of multiple stakeholder communication and the importance of involving all stakeholders when we analyse the requirements of embedded medical software development.

The improvements and suggestions made by each expert have been taken into account, and appropriate changes have applied to the initial version of the proposed process.

In addition to expert review, the Sync-Up process was validated using case studies that were conducted in academic and industrial settings. On the next chapter, the implementation of the process will be presented.

# Chapter 6

# Implementation

## Introduction

This chapter presents the evaluation of the Sync-Up process. The evaluation involves an exploratory and confirmatory case studies that are conducted in academic and industrial settings respectively. The objective of the exploratory case study was to explore the Sync-Up process when developers analyse the requirement by writing user stories and ATs. The confirmatory case study was conducted in three phases. The first phase involves understanding the pre-implementation process of an embedded company. This is followed by the implementation of the Sync-Up process following the proposed phases and generating artefacts. Finally, interviews were conducted to analyse the effect of the Sync-up process. Additionally, archival records and artefacts that were generated from the case study were collected for analysis. The next section presents the exploratory case study.

## 6.1 Exploratory Case Study

The exploratory case study was conducted to investigate the Sync-up process when developers analyse requirements by writing user stories and ATs. To conduct the case study, we set up two teams:

- Team One - composed of four software developers.
- Team Two - composed of a cross-functional team involving three software developers, electronic engineer and embedded engineer.

In both teams, the software developers had previous experience of writing user stories, ATs and following an agile process to produce working software. The teams were given a description of an ideal embedded system project that is described at a high level as shown below:

**Exploratory Case Study Project Description**

A home automation system that controls the room temperature automatically and saves power by switching fans ON and OFF have been presented. For the described system, a temperature sensor will be used to measure the room temperature in real-time. The functional behaviour of the controller has been in a way that when the temperature becomes greater than certain optimum range, the fan should be turned ON and when temperature becomes lower than the optimum range, the fan should be turned OFF.

Both teams were given the project description and were asked to analyse requirements following PHASE:1 of the proposed process, which includes:

- Step 1.1: Define User Stories
- Step 1.2: Prioritise, Estimate User Stories
- Step 1.3: Develop Acceptance Criteria and ATs

- Step 1.4: Develop Release Plan

### 6.1.1 Feedback from Teams

The researcher was involved in observing both teams directly as they followed the steps. All artefacts produced were collected. These included expected artefacts such as user stories, ATs and any extra artefacts, including a list of questions provided by the participants.

#### Team One

The team wrote six user stories of which three were assigned for iteration one and three were assigned for iteration two. Regarding ATs, the team only wrote two ATs leaving the user stories incomplete. The ATs were to check features of the described system. One of the user stories, for example, was *Read temperature*, and the corresponding AT was to check the temperature read value at different time intervals. Quite a few questions were raised by Team One as they did not have direct access to the embedded engineer. Despite developing the user story from the system description, the team was left with additional technical questions such as:

1. What's the source of energy?
2. What's the connection?
3. How often is temperature read?
4. Can we amend/tailor max/min temperature?

In total, there were 15 questions. As these were unanswered, the team had to make certain assumptions and progressed with the understanding that the user stories would need to be revisited once answers were available.

**Team Two**

This team wrote seven user stories from which three user stories were assigned to iteration one and four user stories were assigned to iteration two. The team also wrote four ATs. The user stories that were written by the team also include the basic functional behaviour described in the project description. But this team also wrote detailed ATs as the embedded engineer was collaborating with the software developers.

When this team were analysing requirements, the developers were left to write user stories, and, in some intervals, elaboration/communication was performed. The embedded engineer was able to interact with the software developers and was clarifying issues and confusions that the developers were having. Similar questions have been raised as Team One, but in Team two, the majority of the questions were clarified during the discussion with the embedded engineer. For example, a question similar to one of the questions in Team one, have been raised and the embedded engineer explained:

- How real-time systems work?
- How sensors read data from the environment? and what possibilities are there from the types of sensors.

.

The developers were able to understand the system to be developed with more technical constraints in mind and have two dimensional views of the system. As a result, the user stories were written more easily and clearly characterising the anticipated requirements.

### 6.1.2 Exploratory Case Study Conclusion

The exploratory case study was conducted to explore the benefit of involving technical experts, such as an embedded engineer when analysing requirements. The exploratory case study provides positive feedback in terms of writing user stories, prioritising, estimating and developing ATs. The embedded engineer, as part of a cross-functional team, helped the development teams to understand the technical constraints more clearly and avoid confusions about the system that is going to be developed.

We have also observed that key to effective collaboration is having team members with technical knowledge that will allow the team to understand the concept. For example, one of the participants in Team Two had an engineering background, in addition to software development and was asking different questions allowing the embedded engineer to explain different aspects of the system description.

The main lessons learned from the exploratory case study were:

- A cross-functional team (Team Two) that is composed of software and embedded engineers can write better user stories than a software development team without embedded engineers. The team can easily discuss to clarify technical constraints. On the other hand, the software developers in Team One were only able to write user stories by making some assumptions. Such user stories would need to be revisited if clarifications are made in the future.
- The cross-functional team (Team Two) was able to perform better estimation and prioritisation. The discussion with the technical expert (the embedded engineer) helped the team to understand dependencies and the amount of effort required to complete some tasks of the user story.
- Additionally, a cross-functional team that is composed of software and an embedded engineer was able to write ATs that have more technical constraints.

## 6.2 Confirmatory Case Study

The confirmatory case study was conducted to validate PHASE 1 and PHASE 2 of the proposed process. The case study was conducted in three stages. The first stage involves an exploration of the embedded company's current process. We approached a software architect (former principal software engineer) from the company and conducted an exploratory interview to analyse the previous process. The interview was recorded and transcribed for analysis.

The second phase involves implementing the selected phases. A product team from the company conducted an induction session on the proposed process and, after understanding the steps of the Sync-Up process, the team followed the steps and guidelines suggested by the Sync-Up process when analysing the requirements.

The third phase involves performing a post-implementation interview with the software architect and gathering archival records and artefacts that were generated by the cross-functional team for analysis. The next subsection describes the result of the confirmatory case study.

### 6.2.1 Company's pre-implementation Process

The pre-implementation interview that was conducted with an architect has been presented in subsection 2.3.2. After the interview, a process flow was developed by the researcher and sent to the architect for updates and modifications. The process diagram is shown in Figure 6.1. From the interview we can observe that:

- There were separate teams looking after each different area of a product, i.e. consumer and elite. The embedded development was mostly looked after remotely. Additionally, Sport scientists who act like end users, conduct ATs and make sure that everything is working from their perspective.

- Initially, the development process is driven by the availability of hardware and the embedded firmware interface protocol. The application software development team perform other tasks such as database design and user interface design while the interface protocol is available.
- The implementation of ATs was conducted at the end when the application software is ready. The embedded engineer works remotely and the interviewee suggested that when issues arise during integration stages he prefer to work face to face with the embedded engineer.

### 6.2.2 Implementation

The proposed process was initially presented to the software architect, who was communicating with the researcher. The company was conducting a backlog grooming practice, which involves reviewing the current backlog items and reassessing prioritisation and estimation. During this period the team was willing to evaluate the sync-up process. A cross-functional team (known as the R&D team in the company) that was composed of three product / sports -scientists, the software architect, three developers and three QA engineers were involved during the requirement analysis following the proposed steps.

The R&D team, went through some training on the steps and formats of ATs. The interviewee stated that the team understood the process and agreed to use the process moving forward. The researcher communicated with the software architect and observed the implementation based on indirect observation. The team was working on a product feature for the US Soccer known as drill labelling. It is a feature in the software that allows a user to set predetermined labels to be used when naming drills they cut within the Sonra software. It also allows users to specify primary, secondary and tertiary labels and any combination of these can be used to name a drill.

The user stories and ATs generated from the discussions were shared with the researcher for reference. Sample user stories and ATs developed are shown in Table 6.1

and Table 6.2.

| ID | User Story | Acceptance Criteria |
|---|---|---|
| US_1 | As a user of the coach app, I want be able to create a squad | • Once a user is logged in, they can create a squad<br>• A user should be able to enter data in all fields when creating a squad.<br>• A squad should not be created if required fields are left blank |
| US_2 | As a user, when I create a squad I want to be able to add a profile picture for that squad | • When creating a squad, a user should be able to add a profile picture<br>• When a user adds a picture, the picture should be resized to avoid high res images being added to the system |

**Table 6.1:** Sample User Stories Developed

| ID | Title | Expected Result | Priority | Section | Type |
|---|---|---|---|---|---|
| C3604 | Verify coach can set up a squad | The coach must be able to add a squad successfully | Medium | Squad Management | Functional |
| C3605 | Verify user can join an existing squad | User can receive a notification to join a squad. User can accept the notification and successfully join the squad. | Medium | Squad Management | Functional |
| C3606 | Verify user can add a profile picture for the squad | User can edit the squad by selecting a new profile picture. | Medium | Squad Management | Functional |
| C3612 | Verify user can name their squad | User can name their squad in 2 ways below:<br>1. On initial squad set up<br>2. When squad is edited in settings | Medium | Squad Management | Functional |

**Table 6.2:** Sample Acceptance Tests Developed

After the team finished the backlog grooming activity using the Sync-Up process, an interview was conducted with the software architect to gather the outcome of the requirement analysis based on the proposed process. Additionally, artefacts generated from the discussion, such as user stories and ATs, were collected. The following subsection summarises the result of the interview.

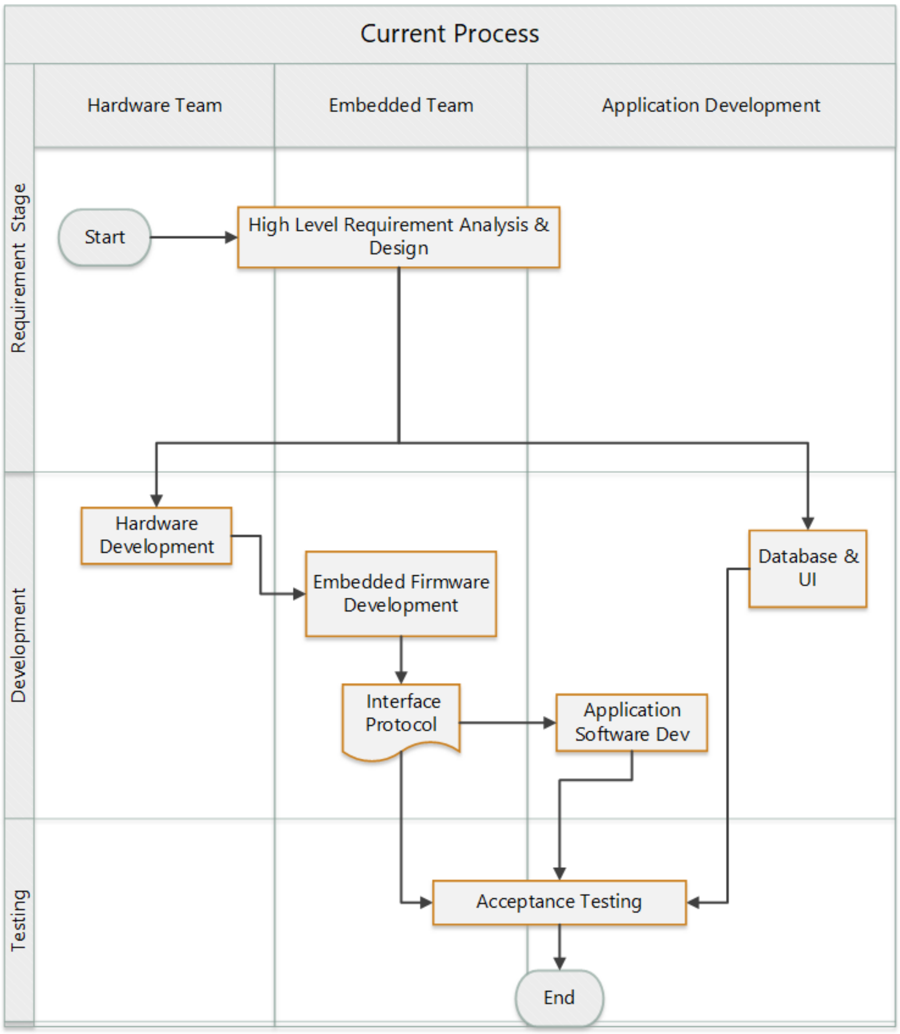


**Figure 6.1:** Pre-Implementation Process Flow of Company A

### Post-Implementation Interview Summary

The interview was conducted using the semi-structured interview questions shown in Appendix G. The interviewee stated that prior to this process the team was analysing requirements through backlog grooming but it was only based on high-level features and epics that would have been broken down into stories and that was as far as it would have went. With the introduction of the proposed process steps, the team conducted the requirement analysis by writing user stories, splitting them out and also writing ATs with the collaboration of the product team and QAs that involves the Sport scientists.

The interviewee stated that using the process, the product team refined the acceptance criteria or definition of done (DOD). The product team then discussed the ATs with the technical members such as software developers. He went on to state that “before the process, we all had our own ATs, but now there’s common ATs from a product team, and everyone was going to align with it”. He stated that done could have different meanings for each of the members involved. For example, done to the firmware developer may mean a working firmware, done to a test engineer may mean all the test are passing and done to the application engineers may mean all the stories are implemented. Having the involvement of all the technical experts syncing up, everyone can agree what’s a DOD for the overall piece of work, not just each individual work.

When asked if discussion between the product team and technical stakeholders brings new information that helps the teams to decide on removing or splitting the user stories, the interviewee stated that prior to the process, coming into the meetings, they were only discussing very high-level epic and user stories. When a story was too big (high-level), they put a lot of points on it, and that usually would need to be reassessed, and the user story had to be broken down into multiple different stories. The interviewee also highlighted that without getting additional information from the product team, they didn’t know how to go about estimating the user stories effectively. He also stated that in terms of prioritisation once they get that additional information through the discussion with the product team, they were able to reassess priorities and got a different value that takes less time.

When asked about the major challenges they encountered when using the process during requirement analysis, the interviewee stated that one of the bigger challenges not just with the sync-up process but any kind of new process involved getting people buying into it and involvement because some people don’t like changes and sometimes it takes a while for people to buy into a different way of doing things. He went on to state that before this process, it was probably a meeting they just attend to understand a high-level user story or epic. The PO would explain features and they would break down into separate user stories and that was it. But after implementing the Sync-Up process, more activities were involved with splitting, estimating, and prioritising the user stories. But also, then the addition of the ATs which would have been lacking before.

When asked if he could suggest a major improvement, the interviewee stated that he would prefer to conduct the Sync-Up the process in a two hour slot. He stated that from his experience, usually cross-functional teams are comfortable in getting the discussions done in two hour periods. But obviously the addition of writing the ATs not just from the product side but also from the QA side, will require more time. The interviewee recommended splitting meetings into separate meetings to avoid team members agitations.

### Summary of Confirmatory Case Study

The confirmatory case study was conducted to validate the proposed process when cross-functional teams analyse requirements. The initial exploratory interview revealed that the ATs were conducted by the Sport scientists and QA engineers after development is finished. Additionally, the team were spending more time as they were writing high-level user stories that required revisiting during later stages. The introduction of the proposed process allowed the product team and sport scientists to collaborate and analyse requirements by writing user stories, splitting user stories, prioritising and estimating the stories and defining a common ATs that will be accepted by all members.

The confirmatory case study revealed that analysing requirements involving all stakeholders will help the team to:

- Write better user stories with detailed information from all stakeholders involving sport scientists and the product team. The discussion also helps the team to split the user stories.

- Conduct better estimation and prioritisation with the involvement of all stakeholders.
- Have common acceptance criteria (DOD) that will be accepted by all stakeholders.

## 6.3 Conclusion of Implementation

This chapter began by describing the exploratory case study that was conducted to explore the proposed process. From the exploratory case study, the expected benefits of analysing requirements syncing up with the embedded engineer were investigated. This chapter also presented the confirmatory case study that was conducted with an embedded company. In the following subsection, the research question RRQ will be revisited.

### 6.3.1 Research Questions Revisited

> RRQ: How can we support multiple stakeholders communication during the requirements analysis stage of embedded medical software development using a combination of suitable agile practices?

In order to address this research question, the research proposed a process using a combination of suitable agile practices that are reported to be used in embedded safety critical domains. The proposed process was evaluated through expert review, that was discussed in the previous chapter.

The proposed process was also validated through exploratory and confirmatory case studies. The exploratory case study revealed that a cross-functional team that was composed of an embedded engineer and software developers develops better user stories with more technical parameters. Additionally, the cross-functional team was able to prioritise and estimate the user stories more clearly. After understanding the expected benefits of syncing up with technical expert such as an embedded engineer, a confirmatory case study was conducted in an embedded company.

Before conducting the case study, an initial investigation of the company's process was conducted using open-ended interview questions. This was followed by the implementation of the proposed process by a corss-functional team composed of sport scientist, product team and software developers. After the team used the proposed process, a post-implementation interview was conducted to evaluate the outcome of the proposed process. The confirmatory case study revealed that analysing requirements involving all stakeholders will help the team to write better user stories, split the user stories effectively, prioritise and estimate the stories more clearly. Additionally, the acceptance tests that were developed with the collaboration between diverse members help the team to have a common DOD that all members will follow.

# Chapter 7

# Summary and Conclusion

This chapter provides a summary of previous work. This is followed by revisiting of the research questions and objectives. Subsequently, the contribution of this research will be presented. Finally, the potential future research will be presented.

## 7.1 Summary and Conclusion

Embedded systems have become an integrated part of our daily lives. These systems span from household applications in appliances, entertainment devices, and vehicles to critical applications. Many embedded systems are safety-critical and may cause severe harm to people and property if they malfunction Bouyssounouse and Sifakis (2005). The development of embedded systems involved the parallel development of hardware and software. A crucial part of embedded systems is embedded software that controls the functionalities of the embedded system.

Within the embedded safety-critical domain, there is an increasing demand for improving the embedded software development process. One approach that may assist is agile software development. The previous report on the usage of agile practices stated that there are some challenges related to embedded safety-critical software development as a result of the parallel development of hardware and software.

To understand the challenges related to agile usage in embedded safety-critical software development and identify agile practices that have been preferred, a systematic literature review was conducted. The review identified challenges and suitable agile practices. Among the challenges identified, a focus was applied and the research focuses on the challenge of multiple stakeholder communication for embedded medical software development.

In order to assist multiple stakeholder communication of embedded medical software development, a process has been developed. The foundation of the process is based on ATDD which has been reported to assist communication and collaboration. Additionally, suitable agile practices from Scrum and XP are also included. Once the initial version of the process is developed, an expert review was conducted to review the process. Leading experts in embedded safety-critical domains and agile software development were involved in reviewing the process, and appropriate amendments were made to address the comments of the experts.

Thereafter, parts of the process were evaluated through performing an exploratory case study in an academic setting and a confirmatory case study within an embedded company. The exploratory case study helps in understanding the expected benefits of syncing with an embedded engineer when analysing requirements. The confirmatory case study conducted in an embedded company helps in evaluating the requirement analysis phases of the Sync-Up process. Two sets of interviews were conducted with the software architect from the company before and after the implementation.

The result of the expert review and industrial investigation have shown a positive result of the process in terms of assisting multiple stakeholder communication of embedded medical software when analysing requirements.

### 7.1.1 Research Contribution

This research aims to provide three key contributions which are:

1. An investigation of the challenges related to agile usage in embedded safety-critical software development domains.
2. Identification of agile practices that were suitable in embedded safety-critical software development and an investigation into how these practices have been used.
3. The development and evaluation of the Sync-Up process to address the issue of poor multiple stakeholder communication during embedded medical software development.

**Contribution to the knowledge of embedded safety-critical community**

A systematic literature review was conducted to investigate the challenges related to agile usage in embedded safety-critical domains. One of the challenges that was identified in the SR was multiple stakeholder communication, which was affecting embedded safety-critical projects. Through the SR, safety-critical domains such as medical & healthcare, automotive, aircraft/avionics, consumer products and general safety-critical reported this challenge. Most of the studies, which are case studies and experience reports, stated that multiple stakeholders in an embedded safety-critical domain have to communicate effectively when analysing requirements to deliver the embedded project in time.

In addition to the SR, the experts that have been involved in reviewing the proposed process stated that multiple stakeholder communication has been affecting the embedded safety-critical companies they were working with. The experts have worked with companies in domains such as medical, flight simulation and different industrial internet of things (IOT). From their embedded project experiences, the experts outlined that diverse teams such as software engineers, firmware engineers, electronics designers and mechanical engineers were working in separate silos that leads to different integration groups. The experts also reported that the embedded projects were affected in their project delivery, wasting developer's time and the need to rework.

### Contribution to the knowledge of agile software development community

Despite having success stories in general software development projects, the SR reported that agile hasn't been used much in embedded safety-critical domains. Prior to the SR, the previous literature covers agile usage and challenges in the safety-critical domain and embedded system themes separately. The SR performed in this research focuses on agile usage in safety-critical embedded software development. The investigation involves the identification of agile practices that have been used and the manner in which the practices have been implemented. The selected studies identified in the SR reported that the embedded safety-critical software development has been looking for rigorous research to advance the usage of agile software development.

Additionally, the experts that have been involved in reviewing the proposed process stated that agile usage in safety-critical embedded software development has been in it's infancy stage and reported that the suitable agile practices that were identified from the SR will benefit the agile community in understanding the practicality challenges of such practices in embedded safety-critical domain.

### The Development and Evaluation of the Sync-Up process

The Sync-Up process is developed through agile practices that are identified from the SR and reported supporting communication. Agile practices drawn from Scrum, eXtreme Programming (XP) and acceptance test-driven development (ATDD) have been used to develop the process. The standard ATDD process has been redefined to involve the additional roles such as embedded engineers, hardware engineers and other technical members to collaboratively analyse requirement and guide the overall development process.

In order to evaluate the Sync-Up process, leading experts in embedded safety-critical and agile software development domains have reviewed the initial version of the process. Recommendations and comments from the experts have been addressed, and the process evolved through several iterations. Additionally, exploratory and confirmatory case studies were conducted to validate parts of the process. The result of the case studies shows a positive result in terms of improving the collaboration of multiple stakeholders when analysing requirements.

## 7.2 Research Limitations and Future Work

There are limitations that should be acknowledged in this research. The first one is that only PHASE 1 & PHASE 2 of the Sync-up process have been implemented in the case studies. The effectiveness of the Sync-Up process would need to be evaluated through the implementation and evaluation of all the phases including PHASE 3, which is the implementation and execution of ATs.

The other limitation of the work conducted is the number of confirmatory case studies. The Sync-Up process was validated through one confirmatory case study. Having more case studies with more embedded companies will help to get more data sources. This would help to validate the the process through cross-case analysis. The research aims to conduct more case studies as part of future work.

Finally the number of experts involved to review the process were not as much as initially planned. Although efforts were made, it was not possible to get more experts to review the process due to availability. To help with this limitation the research focused on getting quality experts. The experts involved in the review process are pioneer in the domain of agile and embedded system design with known publications, books and keynote speech.

In the future, additional case studies will be conducted to validate the remaining phase (PHASE 3) of the Sync-Up process. This phase has not been evaluated due to schedule constraints. As demonstrated through the expert review, the process can be useful to assist the multiple stakeholder communication of embedded medical and other embedded safety critical domains. The future work also includes looking into other embedded safety critical domains. By fully understanding and evaluating overall phases that make the process unique, the fortunes of the Sync-Up process can only improve.

# Appendix A

# Industrial Interview Questions

| ID | | Question |
|---|---|---|
| 1. | | **General Question** |
| | 1.1 | What is your development team made of? |
| | 1.2 | What are you developing? |
| | 1.3 | What have you produced? |
| 2. | | **Requirement Definition and Architectural Design** |
| | 2.1 | How do you define the system requirements of your software and hardware? |
| | 2.2 | Do you involve diverse members (software, embedded, hardware) during requirement design and analysis? |
| | 2.3 | Have you faced communication challenge when different professionals involved in system design and analysis? |
| | 2.4 | How do you define acceptance test (ATs) of the software and hardware? |
| | 2.5 | Are ATs discussed with hardware/embedded engineer? |
| 3. | | **Implementation** |
| | 3.1 | How do you perform implementation of software and hardware? |
| | 3.2 | What hardware related issues have you faced during the implementation of the embedded software? |
| | 3.3 | Do you have to communicate issues with the hardware/embedded engineer? |

3.4 Are you using any dedicated tools/medium to engage with the hardware/embedded engineer?

4. **Testing and Maintenance**

4.1 How do you perform testing and integration of hardware and software?

4.2 Is there a scenario/example where bugs in the hardware affecting the software development process and vice versa? If so, how do you deal with it?

4.3 How do you perform ATs of the software and the hardware?

4.4 Is there a scenario where acceptance testing has been passed from an embedded software perspective and failed from a hardware perspective? What about the reverse?

4.5 How about the interaction between stakeholders during testing?

5. **Risk Management**

5.1 Are you dealing with risks associated with hardware and software?

6. **General Challenge**

6.1 What are the general challenges you faced when you develop software and embedded software?

# Appendix B

# Protocol for a Systematic Reviews (SR)

Surafel Demissie, Dr Frank Keenan, Dr Fergal McCaffery

## Research Questions

The research questions to be addressed by this study are:

- What are the challenges related to agile implementation in embedded medical and safety critical software development?
- What agile practices have been used and how are the practices implemented in embedded medical and safety critical software development?

## Search Process

The search process is based on title, keywords https://www.overleaf.com/project/5ec6ff0df12a8a000125e647and abstract. A set of search strings will be used and aggregated for the outcome from each of the digital libraries that are shown in the following Table:

| **Source** | **Responsible** |
|---|---|
| IEEE Xplore | Surafel |
| ACM Digital library | Surafel |
| Google scholar | Surafel |
| ScienceDirect Surafel | |
| SpringerLink | Surafel |

**Table B.1:** Digital Libraries

## Search strings

```
("agile" OR "scrum" OR "extreme programming" OR "test driven development" OR "lean"
OR "DevOps") AND ("embedded system" OR "embedded software" OR "hardware development"
OR "medical" OR "safety critical") AND  ("challenge" OR "requirement")
```

## Inclusion criteria

Articles on the following topics:

- Studies on agile implementation for embedded software and embedded system development.
- Studies on agile implementation for embedded medical and other safety critical domains.
- Studies that are published between 2010 and 2020.
- Studies that are written in English language.

"Individual researchers (such as a PhD student) can apply a test-retest approach, and re-evaluate a random sample of the primary studies found after initial screening to check the consistency of their inclusion/exclusion decisions."

## Exclusion Criteria

- Studies discussing agile implementation for general purpose software (non-embedded system development).
- Studies that are not in the embedded medical and safety critical domain.
- Studies that are not written in English language.

## Data Collection

Extracted information about the studies should be tabulated in a manner consistent with the review question. Tabulating the data is a useful means of aggregation but it is necessary to explain how the aggregated data actually answer the research questions. The data extracted from each paper will be tabulated based on the following template:

| Data | Description | RQ |
|---|---|---|
| **ID** | Unique identification of each study. | |
| **Source** | IEEE Xplore, ACM Digital library, Google Scholar, ScienceDirect and SpringerLink. | |
| **Year** | The year the study was conducted. | |
| **Author** | Contributing Authors. | |
| **Title** | Title of the study. | |
| **Domain** | The domain of the project or study. | |
| **Study Type** | Empirical study type adapted from Tonella et al. (2007). Study type includes experimental, observational, experience report, case study, systematic review, authors opinion and unclear | |
| **Challenges** | Challenges the study reported. | IRQ.1 |
| **Agile Practices** | Agile practices that have been reported in the study. | IRQ.2 |
| **Summary** | A short summary of the study. | |

**Table B.2:** Data Extraction Template

The data will be extracted by the student and will be checked by the supervisor.

## Data Analysis

Data synthesis involves collecting and summarising the results of the included primary studies. Synthesis can be descriptive (non-quantitative). Using statistical techniques to obtain a quantitative synthesis is referred to as meta-analysis

IT and software engineering systematic reviews are likely to be qualitative (i.e. descriptive) in nature.

### Dissemination

The results of this SLR will be a section of a MSc thesis. The results of this SLR will also be published on a conference paper.

## References

1. Wohlin (2014)

2. Kitchenham and Charters (2007)

# Appendix C

# Selected Studies From SR

| ID | Study |
|---|---|
| P1 | Islam, G. and Storer, T., 2020. A case study of agile software development for safety-Critical systems projects. Reliability Engineering & System Safety,p.106954. |
| P2 | Hanssen, G.K., Wedzinga, G. and Stuip, M., 2017, May. An assessment of avionics software development practice: Justifications for an agile development process. In International Conference on Agile Software Development (pp. 217-231). Springer, Cham. |
| P3 | Jie, J.L.H., 2016. Industrial Case Study of Transition from V-Model into Agile SCRUM in Embedded Software Testing Industries. ACM SIGSOFT Software Engineering Notes, 41(2), pp.1-3. |
| P4 | Bjarnason, E., Wnuk, K. and Regnell, B., 2011, July. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In proceedings of the 1st workshop on agile requirements engineering (pp. 1-5). |
| P5 | Goncalves, G.S., Lima, G.L.B., Maria, R.E., Wisnieski, R.T., dos Santos, M.V.M., Ferreira, M.A., da Silva, A.C., Olimpio, A., Otero, A.G.L., de Vasconcelos, L.E.G. and Sato, L.Y.C., 2015, September. An interdisciplinary academic project for spatial critical embedded system agile development. In 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC) (pp. 8C3-1). IEEE. |

Continued on Next Page

**Table C.1 – Continued from Previous Page**

| ID | Study |
|---|---|
| P6 | Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J. and Oivo, M., 2016, January. Towards DevOps in the embedded systems domain: Why is it so hard?. In 2016 49th hawaii international conference on system sciences (hicss) (pp. 5437-5446). IEEE. |
| P7 | Bjarnason, E., Unterkalmsteiner, M., Borg, M. and Engström, E., 2016. A multi-case study of agile requirements engineering and the use of test cases as requirements. Information and Software Technology, 77, pp.61-79. |
| P8 | Katumba, B. and Knauss, E., 2014, December. Agile development in automotive software development: Challenges and opportunities. In International Conference on Product-Focused Software Process Improvement (pp. 33-47). Springer, Cham. |
| P9 | Gary, K., Enquobahrie, A., Ibanez, L., Cheng, P., Yaniv, Z., Cleary, K., Kokoori, S., Muffih, B. and Heidenreich, J., 2011. Agile methods for open source safety-critical software. Software: Practice and Experience, 41(9), pp.945-962. |
| P10 | Albuquerque, C.O., Antonino, P.O. and Nakagawa, E.Y., 2012, June. An investigation into agile methods in embedded systems development. In International Conference on Computational Science and Its Applications (pp. 576-591). Springer, Berlin, Heidelberg. |
| P11 | Mirachi, S., da Costa Guerra, V., da Cunha, A.M., Dias, L.A.V. and Villani, E., 2017. Applying agile methods to aircraft embedded software: an experimental analysis. Software: Practice and Experience, 47(11), pp.1465-1484. |
| P12 | Lima, G.L.B., Ferreira, G.A.L., Saotome, O., da Cunha, A.M. and Dias, L.A.V., 2015, April. Hardware development: Agile and co-design. In 2015 12th International Conference on Information Technology-New Generations (pp. 784-787). IEEE. |
| P13 | Eklund, U. and Bosch, J., 2012, May. Applying agile development in mass-produced embedded systems. In International Conference on Agile Software Development (pp. 31-46). Springer, Berlin, Heidelberg. |
| P14 | Eklund, U., Olsson, H.H. and Strøm, N.J., 2014, May. Industrial challenges of scaling agile in mass-produced embedded systems. In International Conference on Agile Software Development (pp. 30-42). Springer, Cham. |

Continued on Next Page

**Table C.1 – Continued from Previous Page**

| ID | Study |
|---|---|
| P15 | Hanssen, G.K., Haugset, B., Stålhane, T., Myklebust, T. and Kulbrandstad, I., 2016, May. Quality assurance in scrum applied to safety critical software. In International Conference on Agile Software Development (pp. 92-103). Springer, Cham. |
| P16 | Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T. and Feldt, R., 2014. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical software engineering, 19(6), pp.1809-1855. |
| P17 | Mulder, F.A., Verlinden, J.C. and Maruyama, T., 2014, May. Adapting scrum development method for the development of cyber-physical systems. In Proceedings of the 10th international symposium on tools and methods of competitive engineering TMCE (pp. 19-23). |
| P18 | Douglass, B., 2013. Agile development for embedded systems. In Software Engineering for Embedded Systems (pp. 731-766). Newnes. |
| P19 | McHugh, M., McCaffery, F. and Coady, G., 2014, November. An agile implementation within a medical device software organisation. In International Conference on Software Process Improvement and Capability Determination (pp. 190-201). Springer, Cham. |
| P20 | Martini, A., Pareto, L. and Bosch, J., 2013, June. Improving businesses success by managing interactions among agile teams in large organizations. In International Conference of Software Business (pp. 60-72). Springer, Berlin, Heidelberg. |
| P21 | Alsaqaf, W., Daneva, M. and Wieringa, R., 2019. Quality requirements challenges in the context of large-scale distributed agile: An empirical study. Information and software technology, 110, pp.39-55. |
| P22 | Pernstål, J., Gorschek, T., Feldt, R. and Florén, D., 2015. Requirements communication and balancing in large-scale software-intensive product development. Information and Software Technology, 67, pp.44-64. |
| P23 | Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S. and Kanagwa, B., 2017, September. Requirements engineering challenges in large-scale agile system development. In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 352-361).IEEE. |

Continued on Next Page

**Table C.1 – Continued from Previous Page**

| ID | Study |
|---|---|
| P24 | Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., Könnölä, K., Mäkilä, T. and Lehtonen, T., 2013. Agile methods for embedded systems development-a literature review and a mapping study. EURASIP Journal on Embedded Systems, 2013(1), p.15. |
| P25 | Sekitoleko, N., Evbota, F., Knauss, E., Sandberg, A., Chaudron, M. and Olsson, H.H., 2014, May. Technical dependency challenges in large-scale agile software development. In International Conference on Agile Software Development (pp. 46-61). Springer, Cham. |
| P26 | Duffau, C., Grabiec, B. and Blay-Fornarino, M., 2017, October. Towards embedded system agile development challenging verification, validation and accreditation: Application in a healthcare company. In 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 82-85). IEEE. |
| P27 | Takahira, R.Y., Laraia, L.R., Dias, F.A., Abraham, S.Y., Nascimento, P.T. and Camargo, A.S., 2014, July. Scrum and Embedded Software development for the automotive industry. In Proceedings of PICMET'14 Conference: Portland International Center for Management of Engineering and Technology; Infrastructure and Service Integration (pp. 2664-2672). IEEE. |
| P28 | Shigemura, R.A.L., Goncalves, G.S., Dias, L.A.V., Tasinaffo, P.M., da Cunha, A.M., Mizioka, L.S., Yanaguya, L.H. and Pugliese, V.U., 2018. Using Correct-by-Construction Software Agile Development. In Information Technology-New Generations (pp. 245-253). Springer, Cham. |
| P29 | Heidenberg, J., Matinlassi, M., Pikkarainen, M., Hirkman, P. and Partanen, J., 2010, June. Systematic piloting of agile methods in the large: two cases in embedded systems development. In International Conference on Product Focused Software Process Improvement (pp. 47-61). Springer, Berlin, Heidelberg. |
| P30 | Könnölä, K., Suomi, S., Mäkilä, T., Jokela, T., Rantala, V. and Lehtonen, T., 2016. Agile methods in embedded system development: Multiple-case study of three industrial cases. Journal of systems and software, 118, pp.134-150. |
| P31 | Punkka, T., 2012, October. Agile hardware and co-design. In Embedded Systems Conference (pp. 1-8). |

Continued on Next Page

**Table C.1 – Continued from Previous Page**

| ID | Study |
|---|---|
| P32 | Alhaj, M., Arbez, G. and Peyton, L., 2017, April. Using behaviour-driven development with hardware-software co-design for autonomous load management. In 2017 8th International Conference on Information and Communication Systems (ICICS) (pp. 46-51). IEEE. |
| P33 | Kaisti, M., Mujunen, T., Mäkilä, T., Rantala, V. and Lehtonen, T., 2014, May. Agile principles in the embedded system development. In International Conference on Agile Software Development (pp. 16-31). Springer, Cham. |
| P34 | Liebel, G., Tichy, M., Knauss, E., Ljungkrantz, O. and Stieglbauer, G., 2018. Organisation and communication problems in automotive requirements engineering. Requirements Engineering, 23(1), pp.145-167. |
| P35 | Martini, A., Pareto, L. and Bosch, J., 2016. A multiple case study on the inter-group interaction speed in large, embedded software companies employing agile. Journal of Software: Evolution and Process, 28(1), pp.4-26. |
| P36 | Shen, M., Yang, W., Rong, G. and Shao, D., 2012, June. Applying agile methods to embedded software development: A systematic review. In 2012 Second International Workshop on Software Engineering for Embedded Systems (SEES) (pp. 30-36). IEEE. |
| P37 | Laanti, M., 2016, May. Piloting Lean-Agile Hardware Development. In Proceedings of the Scientific Workshop Proceedings of XP2016 (pp. 1-6). |
| P38 | Test-Driven Development as a Reliable Embedded Software Engineering Practice. |
| P39 | Laukkarinen, T., Kuusinen, K. and Mikkonen, T., 2017, May. DevOps in regulated software development: case medical devices. In 2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER) (pp. 15-18). IEEE. |
| P40 | Mårtensson, T., Ståhl, D. and Bosch, J., 2016, November. Continuous integration applied to software-intensive embedded systems–problems and experiences. In International Conference on Product-Focused Software Process Improvement (pp. 448-457). Springer, Cham. |

Continued on Next Page

**Table C.1 – Continued from Previous Page**

| **ID** | **Study** |
|---|---|
| P41 | Wagner, S., 2014, June. Scrum for cyber-physical systems: a process proposal. In Proceedings Of The 1St International Workshop On Rapid Continuous Software Engineering (pp. 51-56). |
| P42 | Mattei, A.L.P., da Cunha, A.M., Dias, L.A.V., Fonseca, E., Saotome, O., Takachi, P., Gonçalves, G.S., Pivetta, T.A., da Silva Montalvão, V., Kendi, C. and de Freitas, F.L., 2015, April. Nanosatellite Event Simulator Development Using Scrum Agile Method and Safety-Critical Application Development Environment. In 2015 12th International Conference on Information Technology-New Generations (pp. 101-106). IEEE. |
| P43 | Cawley, O., Richardson, I. and Wang, X., 2011, May. Medical device software development-A perspective from a lean manufacturing plant. In International Conference on Software Process Improvement and Capability Determination (pp. 84-96). Springer, Berlin, Heidelberg. |

# Appendix D

# Case Study Protocol (CSP)

**Table D.1:** Case Study Protocol According to Pervan and Maimbo (2005)

| **Section** | **Content** |
|---|---|
| Preamble | The purpose of the protocol is to manage the data collection and analysis of the case study in organised way. The protocol will be stored in secured file storage that will only be shared between involved researchers. |
| General | This research focus on improving the multiple stakeholder communication of embedded teams using a combination of agile practices. The research conducted investigations on agile practices and proposed a process known as Sync-Up Process. The process has been reviewed by leading experts in the domain and modifications and recommendations were addressed. The Sync-Up process has three major phases (Phase: 1, Phase: 2 & Phase: 3). Phase: 1 and 2 focus on analysing requirement while Phase: 3 focus on implementation. With the limitation of time constraint, the case study will aim to evaluate the requirement analysis Phases (Phase: 1 and Phase: 2). |

| | |
|---|---|
| Procedures | To evaluate Phase: 1 and Phase: 2 of the Sync-Up process, the researcher approached companies developing embedded products. The companies should have multiple stakeholders that are composed of for example, product owners, application software development team, embedded firmware/hardware experts, testing experts and other relevant members.<br>Initially a pre-implementation analysis of the companies will be conducted to understand the previous process of the company. Interviews will be conducted with relevant members of the companies. The researcher will give detailed presentations and walk-through of the Sync-Up process. Once the companies agree to evaluate the process, the evaluation of Phase: 1 and Phase: 2 will be conducted.<br>The unit of analysis of the case studies will be the group consisting of multiple stakeholders. The teams will follow the phases of the Sync-Up process when analysing their requirements. Once requirements are analysed following Phase: 1 and Phase: 2, a post implementation interview will be conducted to gather data. Additionally, generated user stories and acceptance tests will be gathers for analysis. |
| Research Instrument(s) | Data will collected from multiple sources. Interview guides and documentation will be used for pre-implementation and post-implementation data collection. The use of more than one data source is a technique known as triangulation that is highly recommended by many researchers Miles and Huberman (1994); Yin (1994); Neuman (2000) as a mechanism for increasing both the reliability and validity of qualitative research. |
| Data analysis guidelines | Detailed description of data analysis procedures, including data schemas, priori codes etc. Based on our hypothesis, the team following the proposed process should develop coherent user stories and AT cases with better quality that will allow the stakeholders to have a common goal and project delivery. |

# Appendix E

# Expert Evaluation Consent

In order to validate the Sync-Up Process, Surafel Demissie, a postgraduate research student at Dundalk Institute of Technology (DkIT), will conduct a series of expert evaluations as part of his PhD thesis. The review will help in any refinements deemed necessary to the process.

I give my informed consent to participate in this evaluation, and I understand that the interview session, which is part of the evaluation process, will be recorded. A summary of the information contributed by me will be included in the final PhD dissertation. I have been informed prior to the review about the precise aims of the review and that the researcher will answer any questions I may have. I understand that my participation is entirely voluntary and that I may withdraw from the interview at any time.

I understand that concerns about any aspect of the review may at any time be directed to Dr Frank Keenan or Dr Fergal McCaffery, supervisors to the researcher at DkIT on:

Dr Frank Keenan: Tel: 0429370200 Dr Fergal McCaffery: Tel: 0429370462

| | |
|---|---|
| ______________________ | ______________________ |
| Participant | Date |
| ______________________ | ______________________ |
| Researcher | Date |

# Appendix F

# Expert Review Questions

## F.1 Section 1

### Background and Experience

1. What is your experience with agile software development and the companies you have worked with?
2. Tell me about your experience with embedded system development?
3. Have you worked with a team of diverse members (software, embedded and hardware) during requirement, design or implementation stages?
4. From your experience, have you experienced with multiple stakeholder communication challenge between diverse members?

## F.2 Section 2

### The Sync-Up Process - Phase 1

1. When defining user stories, do you think syncing/pairing with the embedded/firmware engineer will bring better understanding of the future to be implemented?
2. Do you think syncing/pairing with the embedded/firmware engineer will help in estimating user stories?
3. Do you think syncing/pairing with the embedded/firmware engineer will help in prioritising user stories?
4. When writing ATs, do you think syncing/pairing with the embedded/firmware engineer will help to have better understanding of the ATs of the user stories?

### The Sync-Up Process - Phase 2

1. Do you think discussing with an embedded/firmware engineer or other stakeholders will enable the team to have common understanding?
2. Do you think the acceptance test case that is written syncing/pairing with the embedded/firmware engineer will have better a set of test inputs, outputs and parameters of execution conditions?

### Overall Process

1. Can you name and explain the major benefits you have observed in the Sync-Up Process?
2. Can you name and explain briefly any deficiency you have observed in the Sync-Up Process?
3. Do you have any suggestions to improve the Sync-Up Process?

4. Is there anything else you would like to mention about this process?

# Appendix G

# Case Study Interview Questions

| ID | Question |
|---|---|
| 1. | **General Backlog** |
| 1.1 | What's backlog grooming, and how do you usually do it? |
| 1.2 | Have you involved all stakeholders such as embedded, firmware representative and test engineer when conducting backlog grooming? |
| 2. | **Sync Up Process** |
| 2.1 | Was the involvement of multiple stakeholders affect the team to re-assess the priorities and estimations? |
| 2.2 | During the sync up process, have you got new information that helps the team to decide on removing a user story? How about splitting user stories? |
| 2.3 | During the sync up process, have you got new information that helps the team to create a new user story? |
| 2.4 | During the discuss stage of the sync up process, have you got new information/constraint that helps the team to add information to existing acceptance tests? |
| 2.5 | During the discuss stage of the sync up process, what ambiguities (misinterpretations) of acceptance tests have you clarified? |
| 2.6 | During the discuss stage of the sync up process, what ambiguities (misinterpretations) of acceptance tests have you clarified? |

2.7 Do you think the involvement of technical experts such as embedded/firmware and test engineers will help the team to clarify such ambiguities (misinterpretations)?

2.8 Can we check a random sample of user stories and acceptance test case that shows the changes before and after the sync up process?

2.9 What are the major challenges you encounter when using the sync up process during backlog grooming?

2.10 What changes/improvement will you suggest on the sync up process?

# Bibliography

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002), 'Agile software development methods: Review and analysis'.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2017), 'Agile software development methods: Review and analysis', *arXiv preprint arXiv:1709.08439* .

Alistair Cockburn, L. W. (2000), 'The Costs and Benefits of Pair Programming', *Extreme programming examined,* pp. 223–247.

Anderson, G. and Noessel, &. C. (2017), *Pair-Design Together, Better*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Baheti, P., Williams, L., Gehringer, E., Stotts, D. and Smith, J. M. (2002), 'Distributed pair programming: empirical studies and supporting environments', *TR02-010. University of North Carolina at Chapel Hill Dept. of Computer Science* pp. 86–94.

Barr, M. (1999), *Programming embedded systems in C and C++*, " O'Reilly Media, Inc.".

BCC (2018), 'Latest innovations in medical device technologies', https://www.bccresearch.com/market-research/healthcare/innovations-in-medical-device-technologies.html.

Beck, K. (1999), 'Extreme programming explained: Embrace change', *XP Series* (c), 190.

Beck, K. (2003), *Test-Driven Development By Example*, Vol. 2.

Beck, K. and Andres, C. (2005), *Extreme programming explained : embrace change ,2nd Edition (The XP Series)*, Addison-Wesley.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. et al. (2001), 'Manifesto for agile software development'.

Beizer, B. (1995), *Black-box testing: techniques for functional testing of software and systems*, John Wiley & Sons, Inc.

Berger, A. (2002), *Embedded Systems Design—An Introduction to Processes, Tools, and Techniques s*, Vol. 2002, CMP Books, CMP Media LLC, Lawrence, Kansas 66046.

Bolton, W. (2000), *Microprocessor Systems*, Longman.

Bouyssounouse, B. and Sifakis, J. (2005), *Embedded systems design: the ARTIST roadmap for research and development*, Vol. 3436, Springer.

Brace, I. (2018), *Questionnaire design: How to plan, structure and write survey material for effective market research*, Kogan Page Publishers.

Cohn, M. (2004), *User stories applied: For agile software development*, Addison-Wesley Professional.

Creswell, J. W. and Creswell, J. D. (2017), *Research design: Qualitative, quantitative, and mixed methods approaches*, Sage publications.

Crotty, M. (1998), *The Foundations of Social Research: Meaning and Perspective in the Research Process.*

Dawson. C (2008), 'Pratical research methods: A user-friendly guide to mastering research', *Vasa* p. 153.

Demissie, S., Keenan, F., Özcan-Top, Ö. and McCaffery, F. (2018), Agile Usage in Embedded Software Development in Safety Critical Domain–A Systematic Review, *in* ‘International Conference on Software Process Improvement and Capability Determination’, pp. 316–326.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N. B. (2012), ‘A decade of agile methodologies: Towards explaining agile software development’.

Easterbrook, S., Singer, J., Storey, M.-A. and Damian, D. (2008), Selecting empirical methods for software engineering research, *in* ‘Guide to advanced empirical software engineering’, Springer, pp. 285–311.

Ebert, C. and Jones, C. (2009), ‘Embedded Software: Facts, Figures, and Future’, *Computer* **42**(4), 42–52.

Erich, F., Amrit, C. and Daneva, M. (2014), ‘Report: Devops literature review’, *University of Twente, Tech. Rep* .

Ernst, R. (1998), ‘Codesign of embedded systems: Status and trends’, *IEEE Design & Test of Computers* **15**(2), 45–54.

Fisher, J. A., Faraboschi, P. and Young, C. (2005), *Embedded computing: a VLIW approach to architecture, compilers and tools*, Elsevier.

Forsberg, K. and Mooz, H. (1991), The relationship of system engineering to the project cycle, *in* ‘INCOSE International Symposium’, Vol. 1, Wiley Online Library, pp. 57–65.

Fowler, M. and Foemmel, M. (2006), ‘Continuous integration’, *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf* **122**, 14.

Gärtner, M. (2012), *ATDD by example: a practical guide to acceptance test-driven development*, Addison-Wesley.

Glaser, B. and Strauss, A. (1967), ‘Grounded theory: The discovery of grounded theory’, *Sociology the journal of the British sociological association* **12**(1), 27–49.

Graaf, B., Lormans, M. and Toetenel, H. (2003), ‘Embedded software engineering: The state of the practice’, *IEEE Software* **20**(6), 61–69.

Greer, D. and Hamon, Y. (2011), 'Agile software development', *Software: Practice and Experience* **41**(9), 943–944.

Guba, E. G., Lincoln, Y. S. et al. (1994), 'Competing paradigms in qualitative research', *Handbook of qualitative research* **2**(163-194), 105.

Hannay, J. E., Dybå, T., Arisholm, E. and Sjøberg, D. I. K. (2009), 'The effectiveness of pair programming: A meta-analysis', *Information and Software Technology* **51**(7), 1110–1122.

Hendrickson, E. (2008), 'Driving development with tests: Atdd and tdd', *STARWest 2008* .

Henzinger, T. A. and Sifakis, J. (2006), The embedded systems design challenge, *in* 'Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)', Vol. 4085 LNCS, pp. 1–15.

Hrgarek, N. (2012), Certification and regulatory challenges in medical device software development, *in* '2012 4th International Workshop on Software Engineering in Health Care (SEHC)', IEEE, pp. 40–43.

Humble, J. and Farley, D. (2010), *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*, Pearson Education.

IEC 60601-1 (2005), Medical electrical equipment–part 1: General requirements for basic safety and essential performance, Standard ISO/IEC/IEEE 60601-1:2005, International Electrotechnical Commission.

IEE (1990), IEEE Standard Glossary of Software Engineering Terminology, Standard, Institute of Electronical and Electronics Engineers, Washington, DC.

ISO 24765 (2010), Systems and software engineering – vocabulary, Standard ISO/IEC/IEEE 24765:2010, International Organization for Standardization.

James E. Hewson (2003), 'Cross-functional pair programming | Embedded', https://www.embedded.com/cross-functional-pair-programming.

Janzen, D. and Saiedian, H. (2005), ‘Test-driven development concepts, taxonomy, and future direction’, *Computer* **38**(9), 43–50.

Jeffries, R. (2001), ‘Essential xp: card, conversation, confirmation’, *XP Magazine* **30**.

Jeffries, R. and Melnik, G. (2007), ‘TDD–The Art of Fearless Programming’, *IEEE Software* **24**(3), 24–30.

Kamal, R. (2011), *Embedded systems: architecture, programming and design*, Tata McGraw-Hill Education.

Karlesky, M. J., Bereza, W. I. and Erickson, C. B. (2006), Effective test driven development for embedded software, *in* ‘2006 IEEE International Conference on Electro/Information Technology’, IEEE, pp. 382–387.

Kitchenham, B. A. and Pfleeger, S. L. (2008), Personal opinion surveys, *in* ‘Guide to advanced empirical software engineering’, Springer, pp. 63–92.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J. and Linkman, S. (2009), ‘Systematic literature reviews in software engineering–a systematic literature review’, *Information and software technology* **51**(1), 7–15.

Kitchenham, B. and Charters, S. (2007), ‘Guidelines for performing systematic literature reviews in software engineering’.

Kniberg, H. (2015), *Scrum and XP from the Trenches*, Lulu. com.

Kopetz, H. (2011), *Real-time systems: design principles for distributed embedded applications*, Springer Science & Business Media.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. and Brinkkemper, S. (2016), The use and effectiveness of user stories in practice, *in* ‘International working conference on requirements engineering: Foundation for software quality’, Springer, pp. 205–222.

Marwedel, P. (2006), *Embedded system design*, Vol. 1, Springer.

Mary Poppendieck, T. P. (2003), *Lean software development : an agile toolkit*, Addison-Wesley.

Mc Hugh, M., Cawley, O., McCaffcry, F., Richardson, I. and Wang, X. (2013), An agile V-model for medical device software development to overcome the challenges with plan-driven software development lifecycles, *in* '2013 5th International Workshop on Software Engineering in Health Care, SEHC 2013 - Proceedings', pp. 12–19.

Miles, M. B. and Huberman, A. M. (1994), *Qualitative data analysis: An expanded sourcebook*, sage.

Mugridge, R. and Cunningham, W. (2005), *Fit for developing software: framework for integrated tests*, Pearson Education.

Munassar, N. M. A. and Govardhan, A. (2010), 'A comparison between five models of software engineering', *International Journal of Computer Science Issues (IJCSI)* **7**(5), 94.

Munzner, R. F. (2003), Entering the us medical device market, *in* 'Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439)', Vol. 4, IEEE, pp. 3548–3550.

Neuman, W. L. (2000), 'The meanings of methodology', *Social research methods* **60**, 87.

Palmer, S. R. and Felsing, M. (2001), 'A Practical Guide to Feature-Driven Development'.

Parker, G. M. (2003), *Cross-functional teams: Working with allies, enemies, and other strangers*, John Wiley & Sons.

Pervan, G. and Maimbo, M. (2005), *in* 'Proceedings of the Ninth Pacific Asia Conference on Information Systems', PACIS, pp. 1281–1292.

Pugh, K. (2010), *Lean-Agile Acceptance Test-Driven-Development: Better Software Through Collaboration*, Pearson Education.

Qian, K., Den Haring, D. and Cao, L. (2009), *Embedded Software Development with C*, Springer US, Boston, MA.

Rong, G., Liu, T., Xie, M., Chen, J., Ma, C. and Shao, D. (2014), 'Processes for Embedded Systems Development: Preliminary Results from a Systematic Review', *Proceedings of the 2014 International Conference on Software and System Process* pp. 94–98.

Rottier, P. and Rodrigues, V. (2008), 'Agile Development in a Medical Device Company', *Agile 2008 Conference* pp. 218–223.

Royce, W. (1970), The software lifecycle model (waterfall model), *in* 'Proc. Westcon', Vol. 314.

Royce, W., Bittner, K. and Perrow, M. (2009), *The economics of iterative software development: Steering toward better business results*, Pearson Education.

Rubin, K. S. (2012), *Essential Scrum: A practical guide to the most popular Agile process*, Addison-Wesley.

Runeson, P. and Höst, M. (2009), 'Guidelines for conducting and reporting case study research in software engineering', *Empirical software engineering* **14**(2), 131.

Saunders, M., Lewis, P. and Thornhill, A. (2009), *Research methods for Business Students Fifth Edition*.

Schenk, J., Prechelt, L. and Salinger, S. (2014), Distributed-pair programming can work well and is not just distributed pair-programming, *in* 'Companion Proceedings of the 36th International Conference on Software Engineering', ACM, pp. 74–83.

Schwaber, K. and Beedle, M. (2001), *Agile Software Development with Scrum*, Vol. 18.

Schwaber, K. and Sutherland, J. (2017), 'The scrum guide'.

Shadish, W. R., Cook, T. D. and Campbell, D. T. (2002), 'Experimental and quasi-experimental designs for generalized causal inference'.

Shull, F., Singer, J. and Sjøberg, D. I. (2007), *Guide to advanced empirical software engineering*, Springer.

Solinski, A. and Petersen, K. (2016), 'Prioritizing agile benefits and limitations in relation to practice usage', *Software quality journal* **24**(2), 447–482.

Somerville, M. M. and Somerville, M. M. (2015), 3 – Informed Systems approach, *in* 'Informed Systems', pp. 45–66.

Stapleton, L. (2014), 8.07 – Administrative Evil and Patient Health: A Critique of the Impact of Manufacturing Systems on Health Care, *in* 'Comprehensive Materials Processing', pp. 127–150.

Sutherland, J. and Schwaber, K. (2014), 'The scrum papers: Nut, bolts, and origins of an agile framework (2011)', *SCRUM Training Institute* .

Takeuchi, H. and Nonaka, I. (1986), 'The new new product development game', *Harvard business review* **64**(1), 137–146.

Teich, J. (2012), 'Hardware/software codesign: The past, the present, and predicting the future', *Proceedings of the IEEE* **100**(SPL CONTENT), 1411–1430.

Theocharis, G., Kuhrmann, M., Münch, J. and Diebold, P. (2015), Is water-scrum-fall reality? on the use of agile and traditional development practices, *in* 'International Conference on Product-Focused Software Process Improvement', Springer, pp. 149–166.

Tonella, P., Torchiano, M., Du Bois, B. and Systä, T. (2007), 'Empirical studies in reverse engineering: state of the art and future trends', *Empirical Software Engineering* **12**(5), 551–571.

Vahid, F. and Givargis, T. (2000), *Embedded Systems Design: A Unified Hardware/Software Introduction*, 1st edn, John Wiley & Sons, Inc., New York, NY, USA.

VanderLeest, S. H. and Buter, A. (2009), Escape the waterfall: Agile for aerospace, *in* ‘2009 IEEE/AIAA 28th Digital Avionics Systems Conference’, IEEE, pp. 6.D.3–1–6.D.3–16.

VersionOne.Inc (2017), 11th annual state of agile report, Technical report.

VersionOne.Inc (2020), 14th annual state of agile report, Technical report.

Williams, L. (2010), ‘Pair programming.’, *Encyclopedia of software engineering* **2**.

Wohlin, C. (2014), Guidelines for snowballing in systematic literature studies and a replication in software engineering, *in* ‘Proceedings of the 18th international conference on evaluation and assessment in software engineering’, Citeseer, p. 38.

Wolf, W. H. (1994), ‘Hardware-software co-design of embedded systems’, *Proceedings of the IEEE* **82**(7), 967–989.

Woodward, M. V. and Mosterman, P. J. (2007), Challenges for embedded software development, *in* ‘2007 50th Midwest Symposium on Circuits and Systems’, IEEE, pp. 630–633.

Wynne, M., Hellesoy, A. and Tooke, S. (2017), *The cucumber book: behaviour-driven development for testers and developers*, Pragmatic Bookshelf.

Xie, M., Shen, M., Rong, G. and Shao, D. (2012), Empirical studies of embedded software development using agile methods: a systematic review, *in* ‘Proceedings of the 2nd international workshop on Evidential assessment of software technologies’, pp. 21–26.

Yin, R. K. (1994), ‘Discovering the future of the case study. method in evaluation research’, *Evaluation practice* **15**(3), 283–290.

Yin, R. K. (2003), *Applications of case study research.*

Yin, R. K. (2012), A (very) brief resfresher on the case study method, *in* ‘Applications of Case Study Research’.

Yin, R. K. (2014), *Case study research: Design and methods (applied social research methods)*, Sage publications Thousand Oaks, CA.

Yin and Robert (2009), *Case Study research.Design and Methods.*

Zhang, M., Raghunathan, A. and Jha, N. K. (2014), 'Trustworthiness of Medical Devices and Body Area Networks', *Proceedings of the IEEE* **102**(8), 1174–1188.
**URL:** *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6827202*

Zuill, W. and Meadows, K. (2016), Mob programming: a whole team approach, *in* 'Agile 2014 Conference, Orlando, Florida'.