Testability in an IoT system: a full life cycle perspective

J van Deventer
orcid.org/0000-0001-7217-8594

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Prof JEW Holm

Graduation: June 2023
Student Number: 27062686

Abstract

The Internet-of-Things (IoT) has brought about a major paradigm shift to traditional system analysis and design. System testing, and testability, also changed as more system elements can now be measured than in traditional systems. From a traditional perspective, testability is mostly considered functional testing and verification of devices or components in controlled environments. The massive interconnectivity of systems and devices has thus changed the current testing philosophy, which must be investigated.

This study provides findings from a literature review on the definition of testability in IoT systems, taking into account that a system comprises many elements that can be individually and collectively measured, compared, and actioned. The highest impact on testability, brought about by IoT, is found in the system operations and maintenance phase, including installation, commissioning, scheduled and unscheduled maintenance, and support. In order to consolidate the massive amounts of data, an agent is proposed in a high-level orchestration layer of the system hierarchy. Both literature and a case study confirmed the value of such an agent and its purpose in ensuring system availability.

The impact of IoT on testability was evidenced from a cost analysis (reduced logistics and labour) and reduced resource dependency in an IoT-enabled agricultural irrigation system. IoT had a structural effect on the system resources by changing the resource organisation from a decentralised to a centralised-hybrid structure. In addition to digital twins, a digital agent ensures testability and reflection on the real-world case study confirmed its importance. A hierarchical framework that will ensure the testability of a system in the IoT context resulted from this study.

Keywords: Full life cycle, IoT, Testability

Contents

Acronyms

ADR Action Design Research.

CBM Condition-Based Maintenance.

DSR Design Science Research.

eADR Elaborated Action Design Research.

EOL End-of-Life.

FLC Full Life Cycle.

FR Failure Rate.

ICT Information and Communication Technology.

IoT Internet-of-Things.

MAS Multi-Agent System.

NoT Network-of-Things.

PdM Predictive Maintenance.

QRM Quality Research Management.

RCM Reliability-Centered Maintenance.

RTF Run-to-Failure.

RUL Remaining Useful Life.

RVM Research Validation Matrix.

SE Systems Engineering.

SM Scheduled Maintenance.

SMEs Subject Matter Experts.

TBM Time-Based Maintenance.

VSD Variable Speed Drive.

List of Figures

List of Tables

Chapter 1

Overview

The massively interconnected nature of the IoT has changed how testing occurs. Traditionally, testing occurs during development and production, with operational testing mostly done for diagnostic purposes. IoT improved access to a system's underlying status by providing a comprehensive view of a system, including data on all of its individual elements. As a result, it is necessary to investigate testability in the IoT context, which is the purpose of this research.

The development of IoT systems follows a design process, starting with a need and ending with the final product. This process includes engineering analyses and design, feasibility and economic analysis of the product, and verification testing. The verification stage is where most design processes take very long to complete. The testability of IoT systems up to this point has proven difficult with no definite method or mutually agreed upon way to allow end-to-end testing of these types of systems due to its complexity [1],[2],[3].

The testing of IoT systems often focuses on verifying the system's functionality. However, the testing of the system is not always viewed from a full life cycle perspective [1], which, when done, reveals possible shortfalls of the system which would otherwise not have been detected and therefore not dealt with. Product testability does not end at the place of production but needs to continue throughout the full life cycle up to the decommissioning of the device or system. This poses the following question:

How does one ensure testability in an IoT system throughout its life cycle?

The question is addressed by identifying key focus areas in the system Full Life Cycle (FLC) framework where an IoT system design can take place to ensure testability over its full life cycle.

The study consists of three main sections: a problem analysis; a systematic literature review; and a case study. In the problem analysis, the research topic is broken down into simplified elements to identify specific research challenges to be individually addressed. Three research challenges were identified. The systematic literature review focused research efforts on finding relevant information on the three research challenges and their generic solutions. The literature produced two important findings: (i) information is required to revisit the definition of testability in the context of IoT and larger system status, (ii) that testability is impacted most in the maintenance of IoT systems.

A case study was conducted to find application of the theoretical findings in an IoT-enabled, field-deployed agricultural irrigation system. The case study confirmed that IoT centralises system resources and changes the system resource structure with respect to maintenance. The changed resource structure affected the overall costs of the system, operational (in terms of total downtime) and financial (in terms of maintenance actions and labour hours).

For research to be effective and repeatable, the research methodology must be well-defined. Chapter two captures the definition and discussion of the methods used during this study. The chapter consists of a discussion on Systems Engineering (SE), Design Science Research (DSR), Quality Research Management (QRM), and Elaborated Action Design Research (eADR). The research approach and relevance are discussed in terms of research goals, how they will be reached, and how the research will be validated.

Chapter three provides a problem analysis of the study. Firstly, the research context is created and defined in terms of Systems Engineering (SE). Testability is defined from a SE perspective with specific reference to operations and maintenance. Finally, the research problem is clearly defined.

Documentation of the systematic literature review is presented in chapter four. The chapter begins with exclusion criteria and the text selection process applied in the systematic literature review. Four recurring themes are identified in literature regarding the testability of IoT systems. The scope of research is defined in terms of SE and is limited to the investigation of IoT on system maintenance downtime. A discussion follows on the definition of maintenance from a SE perspective and what IoT looks like in maintenance.

The literature study further focuses on digital twins in IoT. A digital agent is defined in a system context and discussed in terms of its characteristics and implementation as a Multi-Agent System (MAS). The high-level architecture of an IoT system is analysed, and the agent is defined in the context of the study. The chapter finally shows how DSR, eADR, and the digital agent are applied and closed out with a discussion on testability in a system full life cycle.

The case study is presented in chapter five. Chapter five gives the case study design, units of analysis, and data collection process. The case study identified the need for a high-level orchestration layer. Data collection and results are presented, followed by a discussion on the impact of IoT in maintenance. A comparison is made between systems with and without IoT and its impact on testability. Figure 1.1 shows a high-level overview of the parties involved.

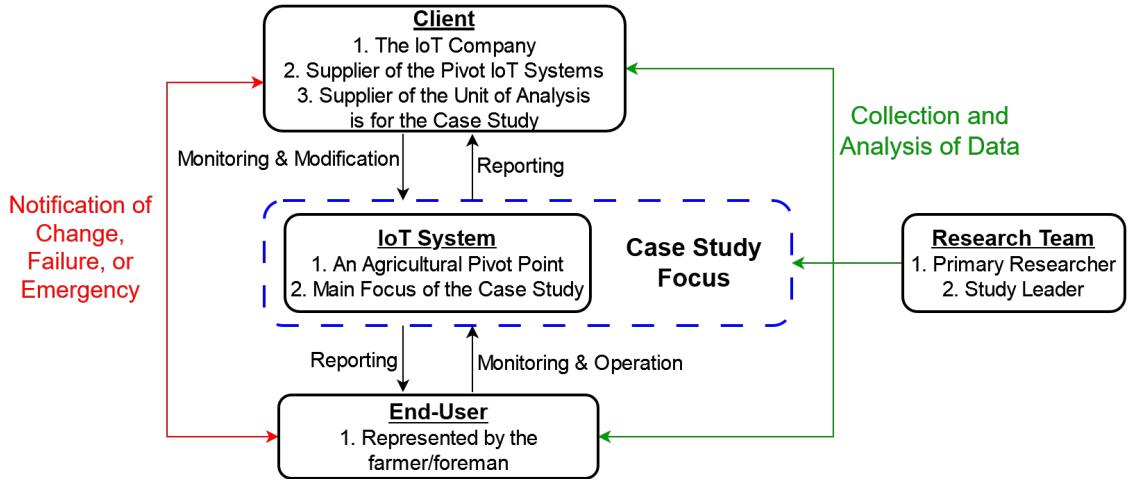

Figure 1.1: Overview of Parties Participating in the Case Study

This chapter also contains an analysis of the resource allocation and organisation in IoT systems (for this case, specifically). Following the resource analysis, an analysis of cost and effort (time) in IoT system maintenance is presented. The chapter ends by showing how testability can be ensured in an IoT system and presents a digital agent in the middleware layer of an IoT system hierarchy.

The thesis concludes with chapter six, where study findings, validation, and future work are presented. A completed Research Validation Matrix (RVM) is presented to validate the research. A generalised framework is proposed to ensure testability in an IoT-enabled system, and recommendations for possible future work are made to conclude this research.

Chapter 2

Methodology

Projects containing research or design elements require a valid method for conducting directed research. Many different research methodologies deliver quality research, which gives many options from which to select. The application of a verified research method ensures that the conducted research is of such a quality that the study or design is reproducible and the results are verifiable.

This study uses a combination of techniques in the DSR paradigm. These methods in the DSR paradigm include SE as a method for real-world system development, QRM for research project management, and Action Design Research (ADR) as the actual research method.

2.1 Systems Engineering

SE is a relatively recent inclusion in the suite of research and development methodologies from a research perspective [4] as it is usually defined as a process guide for the engineering of complex systems [5]. The use of SE has been observed to be more commonly implemented in the field, i.e., in large corporations and industries [4]. This common use of SE is because SE focuses on the system as a whole. The perspective that SE forces the researcher to take is both external to the system (e.g. operating environment) and internal (e.g. engineering design) [5]. This dual perspective of SE provides a general definition of the full life cycle of a system to be discussed and used further in this study. Figure 2.1 shows a general definition of a system's full life cycle for the purpose of this study.

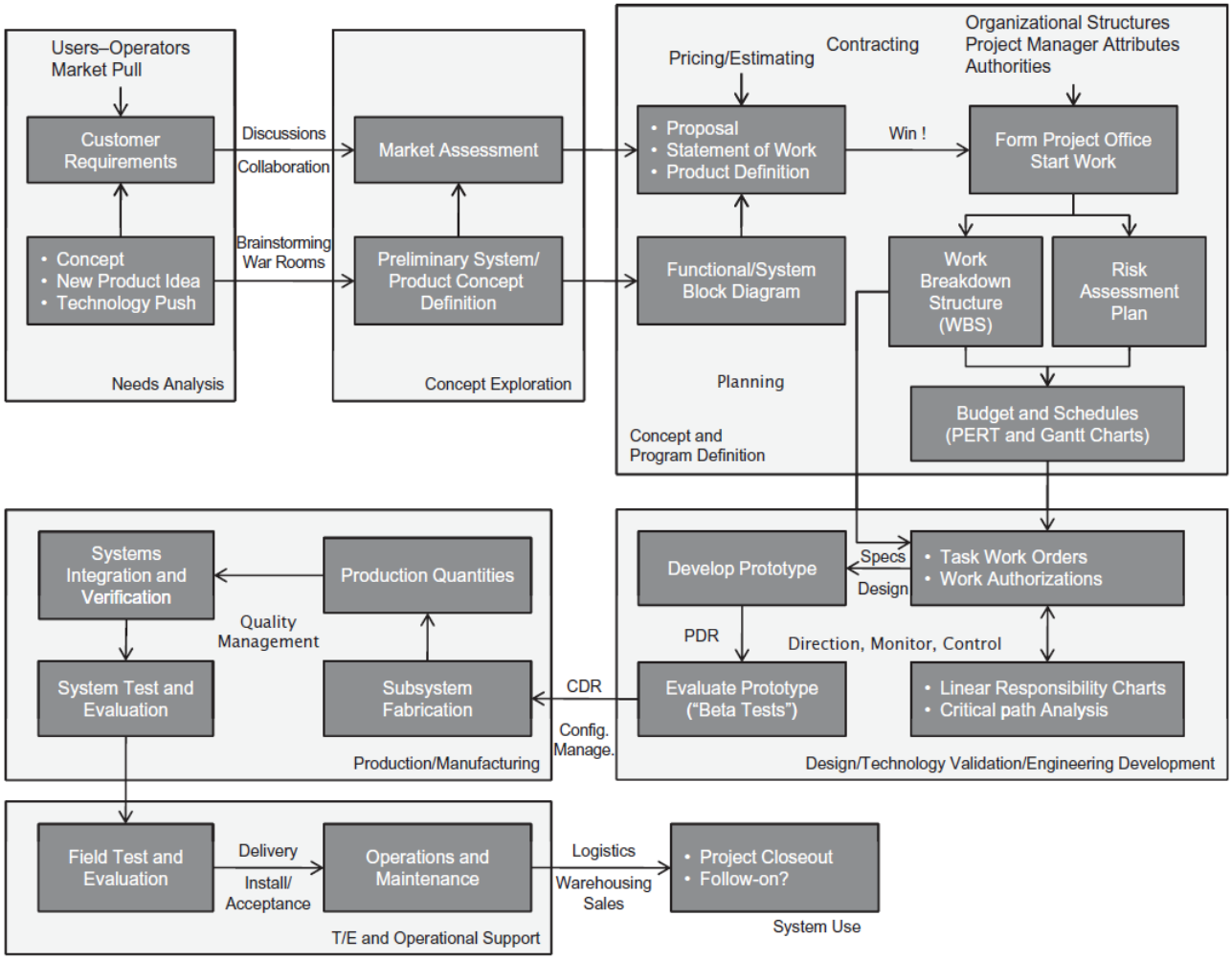

Figure 2.1: General Definition of the Full System Life Cycle, from [5]

2.1.1 Design Science Research

DSR is a common research paradigm inside which action research takes place and focuses on research topics that produce artefacts as deliverables. DSR is commonly used because it enables directed and deliberate research into the topic rather than unconstructed guessing. Gregor and Hevner [6] described this fact very well when they stated: "*The difference between well-conducted research and "hackery", is prediction*". DSR entails the process where a design or research topic is defined, and the outcome thereof is defined (predicted) before the design or research starts [6]. Some DSR methods also call for further design and evaluation stages [6], but in this study, the artefacts are defined and evaluated in a single cycle. Figure 2.2 depicts the DSR process as a research paradigm and framework for design research.

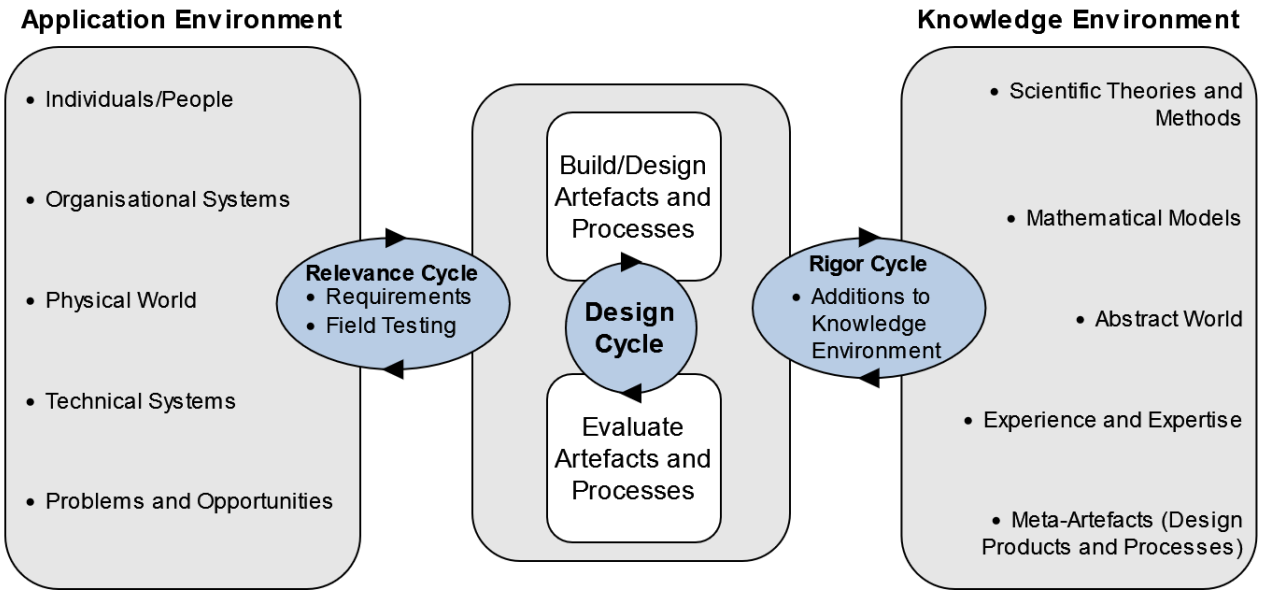

Figure 2.2: Framework Visualisation of Design Science Research, Adapted from [7]

The DSR process provides a link between the physical world where the designed system will exist and the theoretical world where the knowledge base exists [6]. DSR allows the designer to evaluate requirements against characteristics recursively and new knowledge against existing theories to be in line with the requirements from the physical environment. The DSR process thus enables the constant addition of new knowledge to the existing knowledge base, often including newly derived grounded theories (theories based on real-world knowledge). These recursive evaluation cycles are known as the relevance and rigor cycles, as shown in the diagram above.

Incorporating new real-world knowledge occurs during the system's evaluation cycle. The incorporation occurs by utilising a DSR framework, which is well documented in [7] and [8]. Figure 2.2 clearly shows this recurrent interaction between the design, applications environment, knowledge base, and the implementation of the relevance and rigour cycles in the framework.

The use of DSR allows future designs to benefit from the current or past designs. As DSR requires an initial prediction of outcome, the failures in creating an object or system will be well documented if it should not behave as intended from the initial prediction. The reasons why this occurred can then be identified, and solutions for these problems can be addressed in future developments.

2.1.2 Quality Research Management

QRM is a research management method that provides a defined process to ensure the conducted research is of high quality. The QRM model includes important concepts such as change management, requirements management, and traceability.

Generalised Design Process

The QRM framework is based upon a generalised design process as stated in [9]. This generalised process can be divided into six phases, as described below:

Inception and Definition Phase: During this stage, specific definitions must be made. Aspects like the research goal must be defined and verified to align with the expectations of all relevant parties. Research challenges must also be defined to guide the conducted research study. These challenges commonly derive from the research goal and relevant literature. Finally, a research methodology must be defined to ensure the research will follow a verified structure. [9]

Analysis Phase: The completion of a literature study at this stage addressed the research challenges. The research challenges are broken down individually into more minor challenges, which will ease the search for a solution to the main challenge. It is required to document all literature findings relating to the research challenges for use in the synthesis stage.[9].

Solution and Synthesis Phase: This stage uses relevant literature to address the research challenges and propose concept solutions for each defined research challenge. The research method previously decided upon is used here, along with the individual research challenges and their research solutions to design and construct an artefact [9].

Test and Evaluation Phase Results from case studies, experiments, and tests are generated at this stage. The results acquired from valid sources can then be processed and interpreted. The interpreted data can then be used to draw relevant conclusions about the problem [9].

Communication Phase: The documentation of the results occurs at this stage. The results and conclusions are commonly captured in a thesis, article, or academic publication and sent to peers for quality control and review. In some cases, the defence of these reviewed publications will also be produced as public and peer-reviewed publications [9].

Close Out: As the name suggests, this stage is the closing of the research project. This stage is critical as some final adjustments are made and changes managed as final objectives, in some cases, cannot be fully defined in the initial stages of a project [9].

QRM Framework

During the development of a generalised QRM framework, Holm and van der Merwe [9] noted that four main factors would influence the development of such a framework. The identified factors were quality management, systems engineering, project management, and communication. These four factors combined resulted in the development of a RVM, which captured the core of the QRM framework. These factors will not be discussed here as an in-depth discussion on all is given in [9].

2.1.3 Research Validation Matrix

The RVM was developed as part of the QRM framework described in [9]. Holm and van der Merwe defined the RVM as a supporting tool used to manage research efforts in a structured and visual manner [9]. The main focus of the RVM is to ensure that only research that is required and relevant to the subject matter is gathered. The research focus of RVM ensures that resources are well-spent on only necessary research efforts. The implementation of the RVM simplifies the use of the QRM framework described by [9]. It does this by visually including concept solutions, research challenges, and relevant literature into a single matrix while verifying if the literature sources contributed to the concept solutions of the research challenges, hence why it is used in this study.

The RVM includes three sections defined as the following: research challenges, concept solutions, and validation of the concept solutions [9].

Research Challenges

Simplifying the research challenges condenses to "what problems does one want to solve with this study?". The challenges must be supported by relevant research that states the problem to be solved is indeed a problem worthy of investigation. These challenges, as stated previously, need to be derived from the primary research problem the study aims to address [6], [7].

Concept Solutions

During the research phase, where the supporting literature for the research challenges is gathered, it is possible to compile adequate literature that proposes potential solutions to all the identified research challenges. The same research that validates the research challenge could hold the answer to a possible solution to the research problem. These concept solutions are used by the QRM framework during the synthesis phase to create a single solution to the problem.

Validation of the Concept Solutions

Validation of any concept or final solution to the problem can be done in multiple ways. The validation could be done via experimentation, implementation, and observation or by utilising research in the form of case studies where similar situations or problems were identified and addressed.

Implementation of the RVM

The RVM is a compact way to summarise a research project. Figure 2.3 shows a generalised RVM to visualise the concept. In figure 2.3, the left-most column entries are different research artefacts gathered during different stages throughout the research process. The RVM is divided into three sections by the research challenges and the concept solutions. The RVM has a general flow from the top, where the need for research and the problem resides, to the bottom, where the final solution of the problem resides. Arrows in the respective row of each of the research artefacts gathered indicate whether the artefact contributed to the research challenge validation, the concept solution, or both.

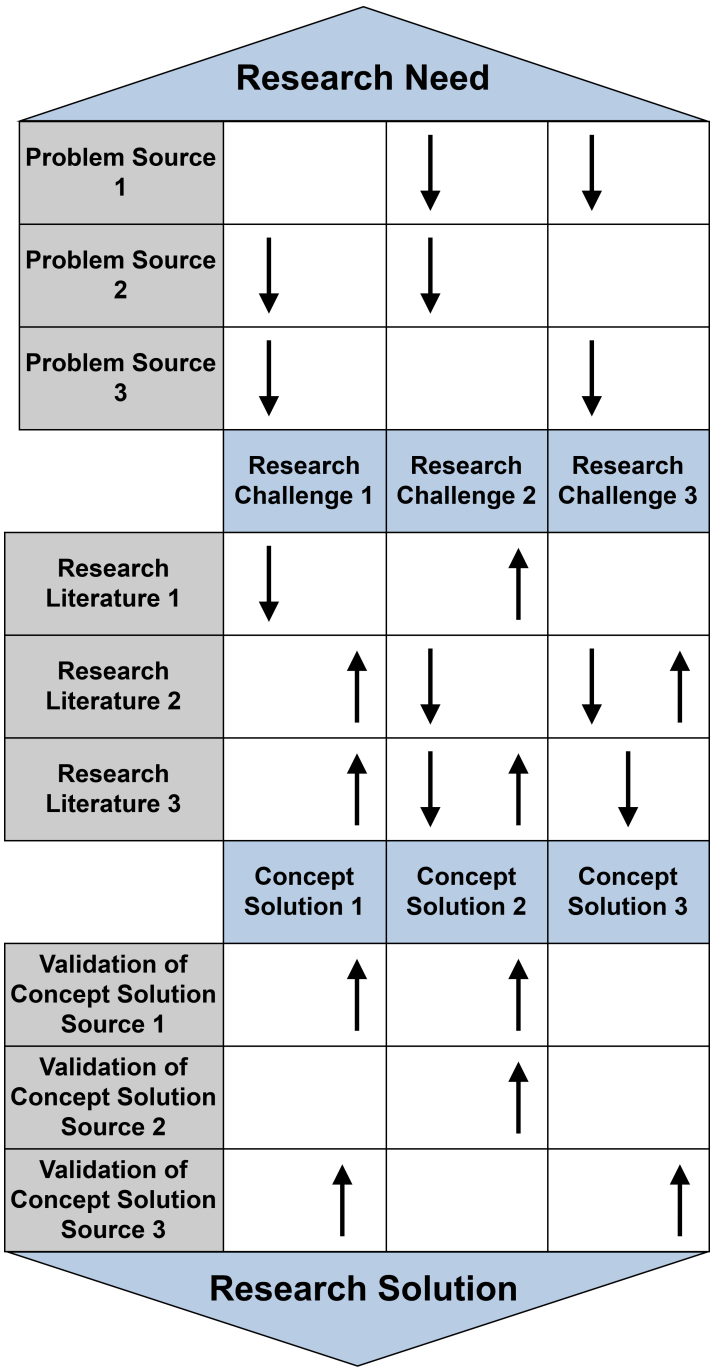

Figure 2.3: Example of a Completed RVM, Adapted from [9]

2.1.4 Action Design Research

ADR is a research methodology most commonly applied to systems or products used in practice and where the researcher forms part of real-world activities. The easiest way to define the process of ADR is by taking action on the current working of the system or product and performing research actions on real-world data. This form of research then requires intervention in the system operation to capture real-world data.

An intervention has to be done scientifically as part of the research, and the ADR paradigm also calls for an initially defined problem and prediction of results. Thus, ADR requires an intervention and DSR requires predicted outcomes, after which the actual intervention can be planned and executed. The intervention is thus planned so that the research results will address the problem. An essential consideration in ADR systems is to assess if a result was coincidental or not [10]. ADR is occasionally repeated in certain situations to ensure any possible coincidences have been eliminated.

Initially, the ADR process was a simple four-stage process. This model was later expanded, and the eADR model was derived and defined as presented in [11].

Traditional ADR

The original ADR framework was defined as a simple four-stage process, each with unique distinct principles representing different phases of a research project. The reason for developing such a process was to develop innovative artefacts in an organisational context and learn from the intervention in the organisation [12]. The traditional ADR framework can be visualised in figure 2.4.

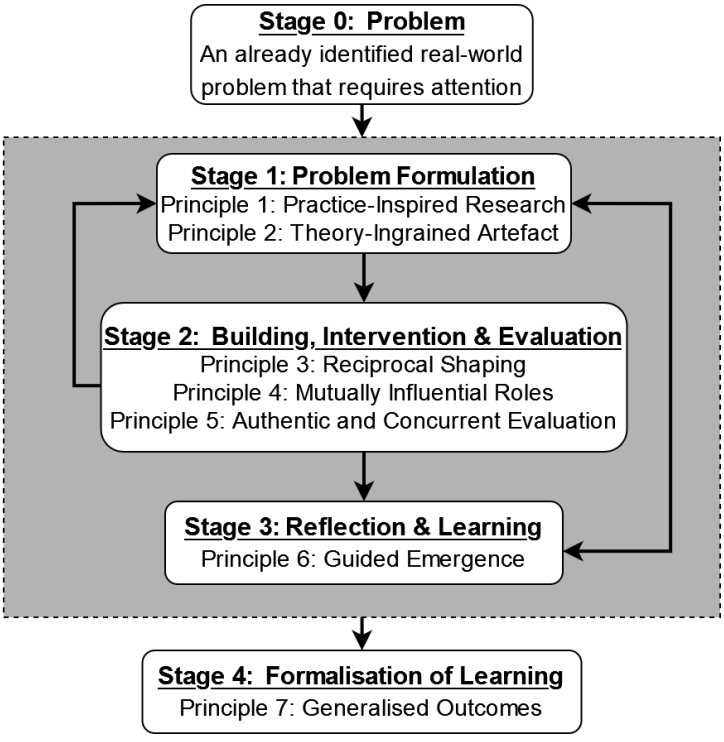

Figure 2.4: Traditional ADR Framework, Adapted from [10],[12]

Problem Formulation: This stage is self-explanatory. It is initiated by identifying or formulating a problem that requires investigation. However, according to the first principle of this stage, the problem must be realised with practice-inspired research[10]. This allows the problem's relevance to be validated and creates an opportunity to create new knowledge in the field. A theory-engrained artefact must be created alongside the practice-inspired research to validate the problem. This is the second principle of the problem formulation stage, as it ensures that the primary informing body of the project is the knowledge base[10],[12].

Building, Intervention and Evaluation: This stage is where researchers commonly spend the most time. This is because this stage needs to be carried out iteratively in a target environment according to [10]. In this phase, the interweaving of the artefact creation, organisational intervention, and evaluation of these individual parts are evident. The outcome of this stage will be realised as a completed design of the artefact [10].

Three principles are embedded into this phase, the first of which is the principle of reciprocal shaping. Reciprocal shaping is based on the idea that there is an influence on the design of the artefact by the increased understanding of the organisational context. This then, in turn, influences the practices in the organisational context [10],[12]. The second principle is mutually influential roles, which focus on interacting and exchanging information between all parties relevant to the research project.

This stage's third principle is known as the principle of authentic and concurrent evaluation. This principle is crucial as, in most research studies, the evaluation process only occurs at the end of the project life cycle. This could result in wasted time and resources, which would be needed again if any problems are found during the research evaluation. ADR compensates for this with this principle as evaluation is recursively done on the research as it is being completed piece by piece, almost eliminating the risk of problems arising later on [10],[12].

Reflection and Learning: This stage in the ADR process could be explained in layman's terms as "recurring verification". This stage runs parallel to the previous two stages and determines which parts of the newly accumulated experience are relevant to be used in the current research project. This stage only contains one important principle, namely guided emergence.

Guided emergence states that all the relevant parties of the research project are actively present and participate in the shaping and creation of the artefact. It also emphasises not only the preliminary design goals and challenges but also any new challenges identified during the recurring evaluation. [10],[12].

Formalisation of Learning: The final stage of the traditional ADR process focuses on the formalisation of what has been learned and emphasises the generalisation of the outcomes [10]. The generalisation is concentrated on the artefact developed during the research process that aims to address a particular problem. Generalisation allows for a similar solution to a broader range of problems. The generalisation is made by consciously moving from specific and unique applications to general, and abstract applications [10]. This generalisation move needs to be done on three levels: Generalisation of the problem instance, the solution instance, and design principles that need to be derived from the design research outcomes [10].

Elaborated ADR

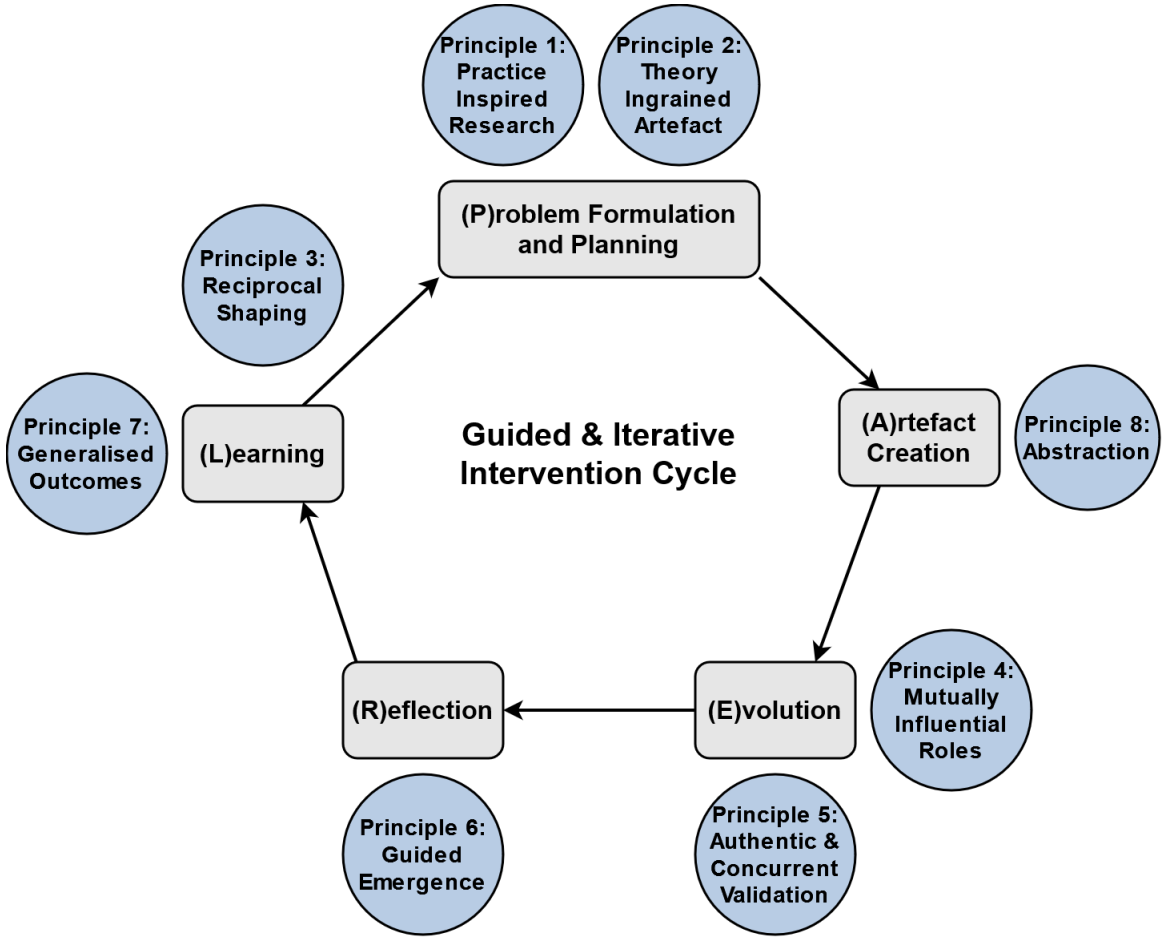

Figure 2.5: eADR Framework, Adapted from [11]

The discussion above shows that the traditional ADR process has been well documented and verified. However, the view of Mullarkey and Henver was that the second stage of the traditional ADR process, captured in section 2.1.4, is mainly open to the interpretation of the reader [11]. They again mentioned that intervention is a core aspect of the whole ADR process and therefore be implemented in every cycle. These researchers reviewed the traditional ADR framework, captured in figure 2.4, and produced what is known as the eADR framework, which can be seen in figure 2.5.

Using the framework from the figure above, Mullarkey and Hevner then reviewed traditional ADR stage definitions. The stages were redefined as the following:

Stage 1: Diagnosis, which included the awareness of the existing knowledge and a clear understanding of the application domain as core principles [11].
Stage 2: Design, which emphasised the identification and conceptualisation of the design of the research artefact that is being created.
Stage 3: Implementation defines the implementation and integration of the completely created artefact into the physical world.
Stage 4: Evolution, which allows for recurrent adjustments to be made as the artefact changes in the defined problem environment.

When eADR is used, it is common to analyse problems and to propose designs without implementation or evolution, which is the case in this study. The reason is that requirements and concepts must often be first defined following a comprehensive process in a complex environment. A valid concept, based on a focused study, provides developers of software with a valuable concept that has been validated, which is very important in the artefact development life cycle. The entry point of the process is thus important, as well as the exit point, which must be defined at the onset of a study.

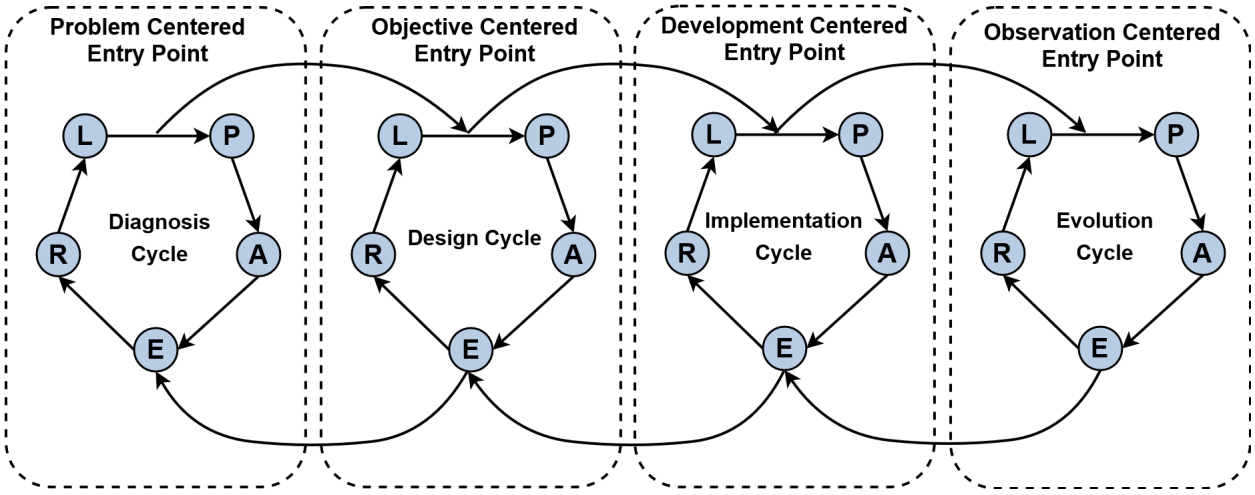

Figure 2.6: Multi-Link eADR Chain Framework with Multiple Entry Points, Adapted from [11]

2.2 Application of the Research Method

Applying the research methodology starts by first identifying the context and current status of all the elements relevant to the study. Our study includes the use of a case study on a client's existing physical IoT systems in the form of agricultural irrigation systems. We also know that irrigation system deployment is the life cycle phase where the main focus will be. By considering the installation, commissioning, and utilisation life cycle tasks as parts of the ongoing system deployment process, the system's IoT context became more focused.

Using SE to define the system life cycle context, all the individual system elements and their interactions could be analysed. SE thus provided a pragmatic way to define and structure the case study in the correct technical context. The use of SE ensured the structure of the study was such that both the high-level system life cycle tasks and system architecture could be placed in context.

The method and paradigm, namely eADR and DSR, symbiotically interacted as eADR provided a method for reflection and expansion of the knowledge base in the DSR paradigm, while the recursive nature of DSR supported a much more detailed technical description for eADR to reflect on improvements and knowledge creation. DSR and eADR were thus used as a research paradigm and method in the clients' SE technical structure.

At the time of this research, the agricultural system provided by the IoT company was in the utilisation phase of its life cycle. Importantly, eADR was used to reflect and to identify inefficient processes in the real-world system technical elements, shortfalls in the installation and maintenance processes, and any possible improvements that could be identified. QRM ensured the focus remained on these objectives, and outcomes were achieved in a timely and focused manner.

2.2.1 Research Approach

The approach to this research can be summarised as outlined in the sections below.

Understand the Impact of IoT on Testability: The first effort, the practise-inspired research stage, focused on collecting information from relevant sources. Research conducted here provided more information by doing the following: (i) the analysis of testing over a full life cycle from a systems perspective, (ii) the completion of a systematic literature review to identify available knowledge, and (iii) the completion of a real-world case study on an irrigation pivot in the context of agricultural IoT to provide a real-world perspective.

Reflect and Learn From the Understanding of IoT on Testability: The guided emergence stage focused on analysis and in-depth understanding of the information collected from the previous stage, as well as additional literature study and case study findings (the literature study did not terminate at the end of the first stage). The information was used to identify where IoT would add value to the client's already existing system full life cycle, focusing on operational and maintenance elements. Learning from the literature review and the case study, how IoT will affect testability would become evident.

Define a Framework to Ensure Testability for the Client's Real-World System: This stage focused on applying gathered information to create an artefact in the form of a concept framework to understand and check for testability from a full life cycle perspective. Areas were identified where, in the maintenance system of the client's IoT system, value could be added to ensure testability. An approach was thus developed to support testability following from the study in the prior stage.

The above stages aligned with the QRM management phases and were managed accordingly.

2.2.2 Relevance

The research approach had to address a real-world problem, which required relevance. This was achieved by establishing clear links between the real world and the theoretical world, as well as verifying findings from research in practice, as discussed below:

Relevance - Alignment with the Real-World: DSR allowed relevant artefact to be defined by linking the real world to the theoretical world of IoT systems [6]. SE further ensured real-world relevance from system analyses and modelling according to system function, form, and fit [5]. Hence, the need for relevance was addressed.

Verification - Case Study Research: A case study provided a clear understanding of how IoT impacts testability in the real world. The case study was also used to verify the system testability concept. In line with good case study principles, observations from the case study were used to learn, conceptualise, and verify how testability is affected by IoT principles [13]. eADR was used in the case study as a valid method for interacting with the real world while learning and improving [10],[11],[12]. Analysis and reflection as applied in eADR were also addressed in the structured case study.

2.3 Conclusion

This chapter provided a definition of the DSR as the research paradigm, eADR as an action research method, SE to provide a technical framework and context, and QRM to ensure a structure research management approach was followed. Application of these tools was discussed in the context of the client's agricultural system, and real-world relevance was shown to be achieved using DSR, eADR and case study research.

Chapter 3

Problem Analysis

The research problem was divided into research challenges that could be addressed individually. Generic solutions to each challenge were compiled, and a systematic approach was initiated to generate more detailed solutions to the generic solutions. This chapter discusses the research challenges that were identified in this research project.

3.1 Creating Research Context: The Full Life Cycle

Because of the importance of the full life cycle, it was decided to include an overview of the life cycle in this section, as all consequent discussions will refer to this cycle as a first step in the problem analysis process. Importantly, IoT provides massive interconnectivity, vast amounts of data, and centralisation that will change the research context significantly. The discussion below is thus to (i) validate the application of the above research philosophy and (ii) provide important context to the research project. Shown below is the well-known life cycle, followed by discussions of each phase and the influence of IoT in the sections afterwards.

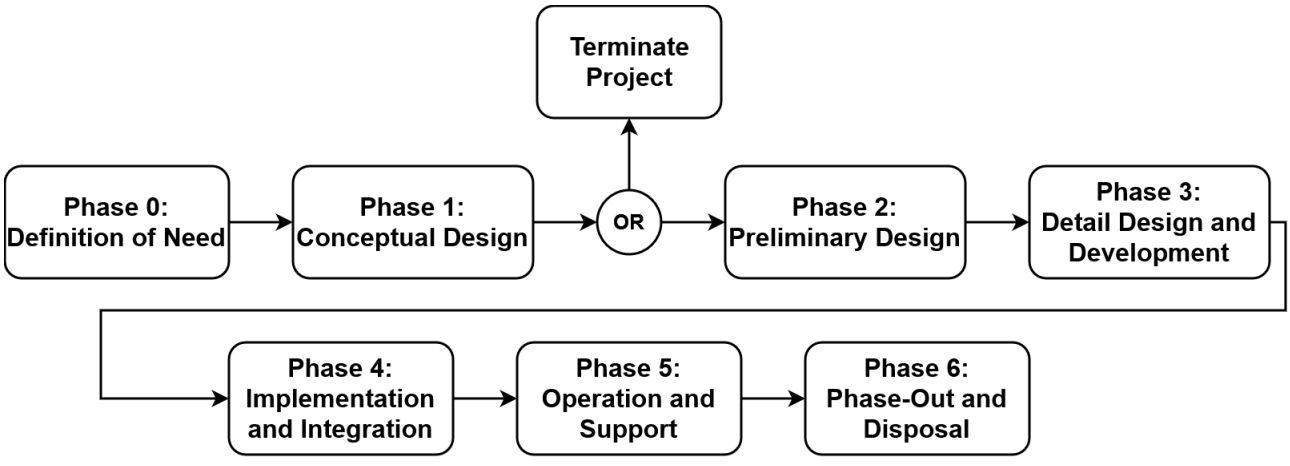

Figure 3.1: High Level IoT System Life-Cycle

3.1.1 Definition of Need (Phase 0)

The definition of requirements must consider the system context to understand the impact of IoT as the main advantage of IoT is to provide connectivity. The high-level needs analysis must thus consider the ability to obtain system status as design input. For example, a hazardous environment will limit the on-site testability of the system, whereas a network-isolated environment will make an IoT component completely unusable. Specifically, interfaces between IoT devices, the IoT network, and centrally located control room entities must be used to guide the requirements elicitation process. Access to information implies that information management is a crucial consideration.

3.1.2 Conceptual Design (Phase 1)

Testability is a design guideline and requirement that will require a change to the traditional approach in the IoT context. That is, the availability of system status will not only allow near real-time testability of the overall system (its FULL status) but will affect operations and maintenance systems in various ways, including the ability to perform comprehensive centralised monitoring and evaluation of a system's overall health status. Therefore, this system characteristic must be included in IoT system design as it will inform maintenance planning and workforce management in practice, support system resilience analysis and optimisation, and ensure system availability in general. Here, also, IoT system concept design must consider the availability of vast amounts of information and must ensure information management infrastructure is included in the design.

3.1.3 Preliminary Design (Phase 2)

The focus of the preliminary design is to perform system modelling and performance evaluation and to define system development specifications. Testability modelling will focus mainly on the system's maintenance and support components, and information availability will improve overall system maintenance effectiveness factors. Availability of system status data, and its dynamics over time, will allow an analyst to perform more detailed failure scenario planning when combined with failure mode analysis and related analysis methods. Here, the application of data analysis techniques could be a consideration, and appropriate sub-systems are typically designed and evaluated to support advanced data acquisition and analysis in practice.

3.1.4 Detail Design, Integration and Testing (Phase 3)

The detail design phase implements testability according to development and product specifications, with a firm reliance on testing of the system artefacts. That is, testability also (still) includes verification and validation of product functional, physical, and interface requirements. This research focuses more on the operational and maintenance aspects of IoT systems as treated in the case study. However, the overall system testability will include at least functional testing. In an IoT environment, the testability of system elements in the production process is also evaluated as such tests require interconnectivity.

3.1.5 Implementation and Integration (Phase 4)

Deployment requires specific attention to asset tracking, installation success factors, and commissioning data (cold and warm). A product installed as part of a larger IoT system will be interconnected, requiring testability of interconnections (capability and performance) and product-specific verification testing. During operation, IoT products ("things") will be connected to sub-systems and larger systems.

3.1.6 Operation, Support and Maintenance (Phase 5)

Implementing IoT can potentially include convenience and cost reduction elements in the operations phase. The system can be controlled, monitored and, to a degree, maintained from a centralised location. This full remote access to the field system allows a client and a central system to control field systems at a moment's notice, whereas control without IoT integration would require travel time and other logistic challenges. From a manufacturer's perspective, the IoT connectivity allows for remote access that would reduce labour-hours of a manufacturer as well as downtime for a client.

3.1.7 Phase-out and Disposal (Phase 6)

From asset management and environmental responsibility perspectives, the life cycle data must be recorded and performance history (including failure modes) be analysed for future decision-making. This data may be used to support the planning and design of future systems. Recycling, reconditioning, and disposal data will result from the phase-out in order to close out this phase. This data is typically used for regulatory compliance.

In the following sections, more detailed analyses of the problem are done by focusing on specific aspects of IoT and its application in testability. This will focus the discussion and research, given the broader context provided above. More formal definitions are provided for this research.

3.2 What is Unique to an IoT System?

The IoT can be defined in different ways depending on the researcher and the context. According to [14] IoT can be defined as "*a type of network to connect anything with the internet based on stipulated protocols through information sensing equipment to conduct information exchange and communications in order to achieve smart recognitions, positioning, tracing, monitoring and administration*". According to P. Ryan and R. Watson, the IoT can be defined as the following: "*The Internet of Things (IoT) is an extension of the Internet in which large numbers of "things", including sensors, actuators, and processors, in addition to human users, are networked and able to provide high-resolution data on their environment and exercise a degree of control over it*" [15].

These definitions, although different, are quite similar in the information they convey. In layman's terms, we now understand that the IoT is a large network of multiple different hardware, software, and human elements implemented to generate, collect, and process data. Following this definition, different characteristics were identified, collectively describing every aspect of an IoT system. Some of these characteristics are exclusively applicable to IoT systems, and thus we can accept that these characteristics are what differentiates IoT systems from others. These fundamental characteristics are listed below along with a brief description of each [14],[16],[17],[18],[19]:

- **Interconnectivity:** which simplistically states that anything can be connected to the local or global network.
- **Things-Related Services:** which states that any node in the network classified as a "thing" can provide services such as privacy protection or consistency between different physical things and their virtual counterparts.
- **Heterogeneity:** states that different nodes on the network do not have to be based on the same hardware or physical thing that it connects to the network. Different hardware platforms can interact with each other or service platforms over a wide range of networks.

- **Dynamic Changes:** which describes the ever-changing world of IoT as technology develops exponentially and hardware platforms that connect things to the network will have to be upgraded or replaced, which changes the network configuration. Another change is the constant changes in the states of the nodes on the network, such as location, connection speed, and connection status.
- **Enormous Scale:** which describes the ever-growing size of the IoT and the related sub-networks encapsulated in the larger networks.
- **Safety and Security:** which defines the responsibility of IoT providers and network designers to ensure the local sub-networks are properly secured end-to-end and the data transferred on the network is secured from outside access, and no personal data is distributed without proper consent.
- **Connectivity:** which defines the ability to access the network by devices or operators, the compatibility of the network with other networks and devices, and access to the data being transferred on it.

3.2.1 What is Testability in terms of IoT?

Considering the general definition of testability, we may argue that the underlying principle of testability may have to be revisited in the context of IoT. According to [20], testability is defined as the ability to run an experiment to test a hypothesis or theory. With this general definition, still applying it directly to any IoT system is difficult. In any IoT system, multiple sub-systems exist and, in some cases, countless nodes that could act as testable points. Research was done into how testability in an IoT context is currently defined.

According to [1], testing an IoT system is challenging when using physical hardware. The reason being that many widespread hardware nodes are needed to allow for a real-world test. Testing wireless networks proves similarly difficult as it requires isolation of wireless connections between equipment and the surrounding environment. In the IoT world, the competitive side of the industry is located in the software provided along with the hardware, as basic node hardware is a commodity and is hard to differentiate as a product advantage in the market [1]. This has resulted in the scope of testability of IoT systems to be primarily focused on the software component. Figure 3.2 shows the scope wherein testability is defined.

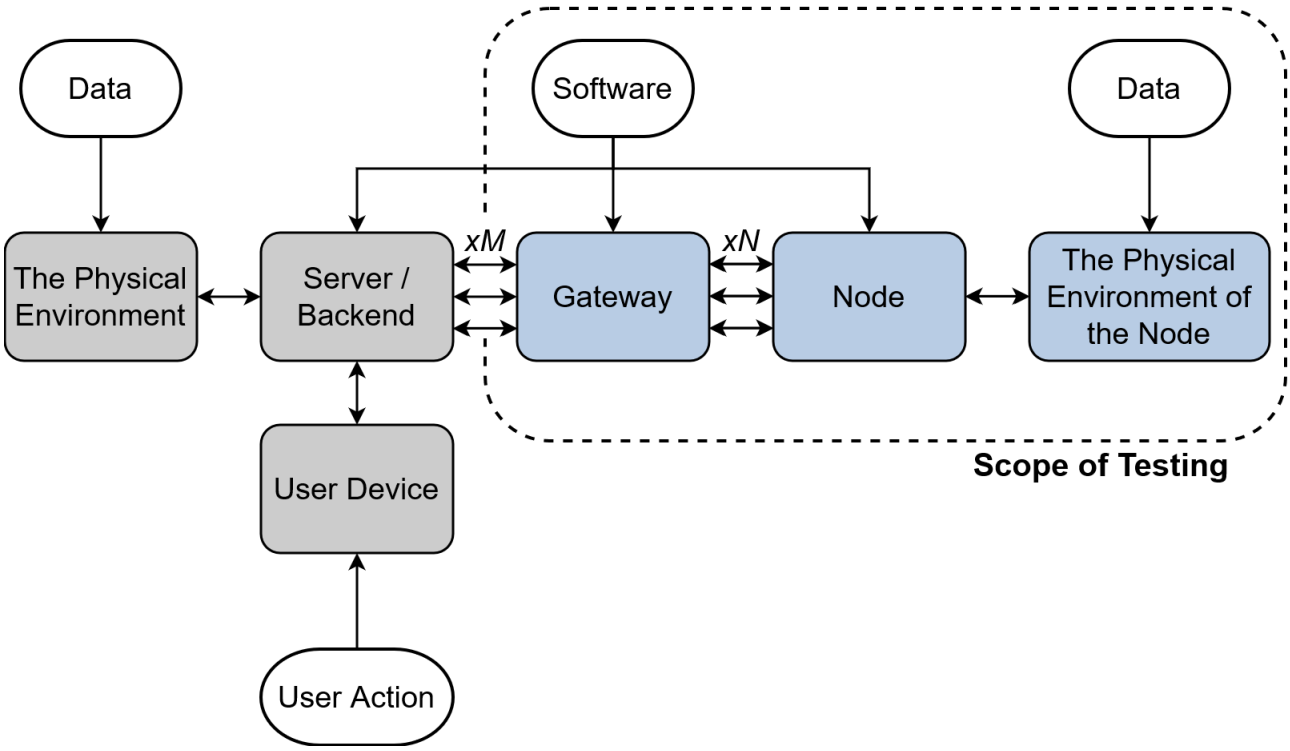

Figure 3.2: Scope of Testability in terms of IoT, Adapted from [1]

3.2.2 How Does Testability Fit Into a Full Life Cycle?

The full life cycle of any device or system is not always considered when a product is being developed. Based on the general discussion above, a challenge arises to clearly define test activities (testability, in general) in each phase. This aligns with the question: if one does not have an expectation of the outcomes of a test, what good was the test? Or as [3] puts it, "*if you do not know if an output is correct after a test is performed, what is the point of testing?*"

A definition of testability from [21] assists in this context. According to [21], testability addresses the extent to which a system or unit supports fault detection and fault isolation in a confident, timely, and cost-effective manner. This definition better explains that testability should be seen not as a physical process or experiment that is conducted, but as a characteristic that is incorporated into the design of a device or system. With these definitions in mind, this study will further investigate and review the general definition of testability and will revisit what testability in terms of IoT from a full life cycle perspective means.

3.2.3 Is It Possible to Fully Test an IoT System?

There is still no definitive answer to the above question, but sufficient information exists to answer it for our purposes. To answer the question, it could be beneficial to first ask if it is possible to test the internet, which is highly unlikely. According to [3], the same applies to IoT.

It is, however, possible to thoroughly test sub-nets of the IoT or individual nodes as these are bounded components encapsulated in the whole of IoT. The solution suggested by [3] involves the definition of something called the Network-of-Things (NoT). The NoT defines bounded components of the IoT, allowing practical testing methods to be implemented. Depending on the size of the IoT system that needs to be tested, the concept of NoT could be implemented by either classifying the whole system as a network or breaking the whole system down into smaller networks.

3.2.4 How Does the Size of an IoT System Affect Testability?

As mentioned in section 3.2, enormous scale is one of the main characteristics of an IoT system. The scale of the IoT and its sub-networks is ever-growing and will continue for as long as technology evolves. This raises concerns as the testability of an IoT system is already complex, as stated in section 3.2.3. The question then becomes not whether the size of the IoT network affects the testability thereof, but rather how the size affects the testability thereof.

According to [3], [22], large-scale NoT systems are likely to process and produce large amounts of data. The data produced and processed is likely to be for the purpose of making limited decisions such as 'activate' or 'deactivate' [3]. This limited decision-making capability of the network creates a difficult testing situation where the data cannot be validated to determine if internal errors or data corruption occurred [3].

From a physical perspective, the testability of an IoT system with a large-scale network will rely more on the simulation approach to testing. In section 3.2.1, the difficulties regarding the hardware testing of an IoT network were discussed. If these difficulties arose while testing small-scale IoT systems, it would prove highly complex and resource-intensive to perform hardware testing. Therefore, a well-modelled simulation would be required.

3.2.5 What is Testability in terms of the SE Full Life Cycle?

Given the broader context and the more focused discussions that followed, it is possible to revisit the definition of IoT testability over a full life cycle.

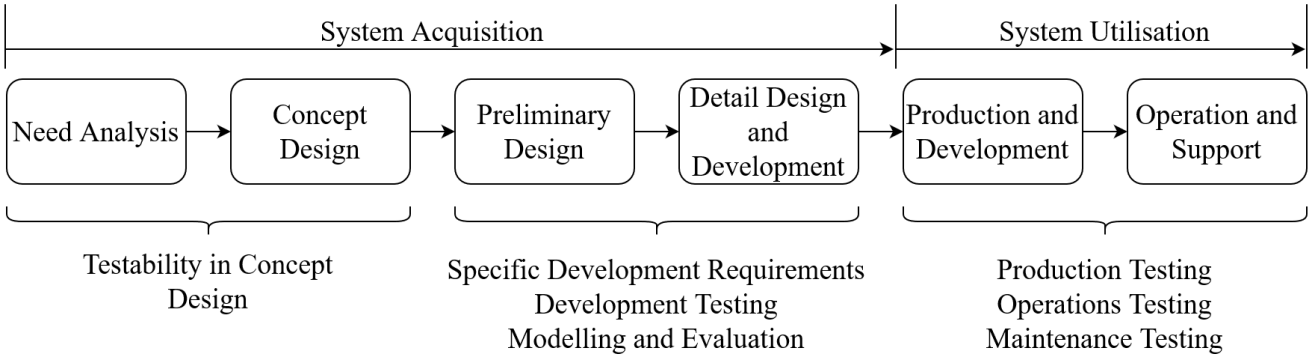

Figure 3.3: Testability in the Full Life Cycle, from [23]

During this study, original work was completed and published in the form of an academic article, captured as appendix A, and used throughout this study. The article, being original work, will not be referenced. In the simplified diagram shown in figure 3.3, it is thus clear that testability forms part of each phase of the process in some way or another, be it during planning, designing, or utilising. There is a need to consider testability at the later stages of the life cycle in order to correctly define and specify what must be done in the initial stages of the process. With the traditional focus of testing, being production and diagnostic testing, and it being defined and contained in a controlled environment, the focus of the research shifts towards field operation and support where testing is evident in the utilisation phase of the life cycle.

By now, it is evident that testing can take many forms. Requirements, design references, model evaluation, and detail system testing are all verified in the SE process, which is also a form of testing. In the high-level operation and support tasks, the importance of the real-time overall system status becomes critical for timely failure monitoring, system performance management, and system performance management.

This will be the focus of our research since the IoT concept shows more value in the utilisation phase, with specific attention to operations and maintenance.

3.2.6 What is Testability in Operation and Maintenance?

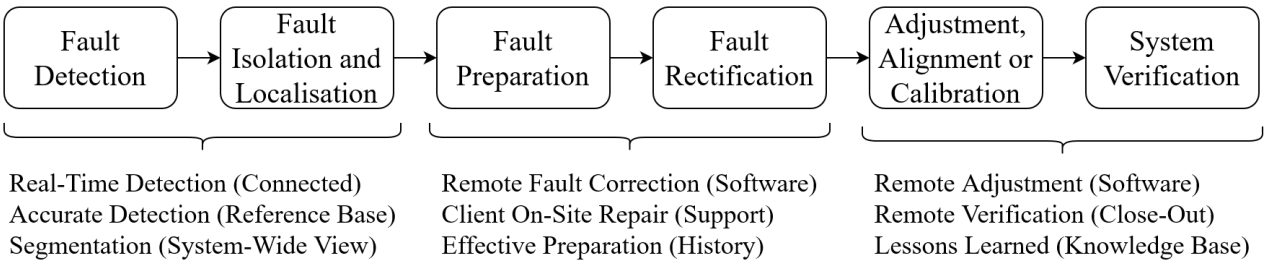

Figure 3.4: The Corrective Maintenance Cycle and IoT Related Activities, adapted from [23]

In figure 3.4, the corrective maintenance cycle obtained from literature was reduced to six high-level tasks as can be seen in [23]. During each of these tasks, IoT will have an impact in such a way as to improve each phase's efficacy or efficiency.

1. Fault Detection: The system will be able to detect faults either proactively (by determining if the system deviated from the expected value) or reactively (by determining if a functional capability error occurred) by using remotely obtained data from the system regarding its status. Historical data being available and can be used to identify predefined faults (by means of pattern recognition) and report on functional or operational exceptions (anomaly detection).

2. Fault Isolation and Localisation: The IoT enabled system will be able to isolate faults from a system point of view, as opposed to a functionality (product) point of view, by utilising data captured from the different nodes and interconnections on the network. Historical data will also be available as system logs that will support the system view as actionable information.

3. Fault Preparation: Fault preparation can be done by utilising historical operation and maintenance data, environmental data, and system logs to determine the maintenance requirements for the system. The data can also be used to characterise the system to determine if specific preparation actions are required.

4. Fault Rectification: With the addition of an IoT component, fault rectification can take place remotely in most cases. With the availability of the system status and the IoT connection, subject experts are able to instruct and advise rectification actions remotely according to the real-time data reported by the system. This is made possible by the remote control functionality and diagnostic data the IoT component provides.

5. Adjustment, Alignment and Calibration: Adjustment may be made in either a proactive (reconfiguration of the system after an update) or reactive (changes in the configuration after an unexpected external change occurred) manner based on system status data provided in real-time. The inclusion of IoT makes the provision of real-time data possible. System verification is done similarly.

When testability is considered as the ability to obtain the status of a system at any given point in time and to compare the status data against a valid predefined reference, it becomes possible for condition (resource integrity) testing and performance (functional capability) testing to take place.

3.3 Formal Problem Definition

Clearly, the research problem that will be addressed in this study is complex and consists of multiple complex elements. To formally define the problem statement, the main components of the research topic can be addressed first. Defining what exactly sets an IoT system apart from any other was the first step in the process. Since it was found that IoT has a fluid definition depending on the context, creating difficulty in defining what is unique, the unique characteristics of an IoT system were identified and used as the differentiating factors.

Defining what testability is in terms of IoT proved difficult as no clear information is available on testability in the broader IoT system context. Different opinions and definitions exist in the general sense, but these definitions are focused more on functional testing. How testability fits into the full life cycle similarly had a very narrow scope as it was localised to the implementation and installation stage of the life cycle.

Although the scope of testability should be increased to include the whole life cycle, functional testing still plays an important role. Determining if an IoT system of any size is testable and how the size affects the testability thereof was vital information to gather when the life cycle is to be investigated.

These individual elements were uncovered while conducting research towards the greater goal of this study, what is the impact of testability in an IoT system when considering the full life cycle. By the use of literature research, observations made from industry applications, and the information gathered during the case study, the research challenges could be properly defined.

The research challenges that were identified are as follows:

1. No formal definition exists for testability in terms of IoT.
2. No clear information is available on how testability impacts the full life cycle.
3. No clear information is available on what the full life cycle looks like when viewed from an IoT perspective.

At this point, QRM requires a definition of the research problem in terms of individual research challenges. The top section of the RVM (described in section 2.1.3) shows the problem definition. Figure 3.5 gives the research challenges and the source from which each challenge had been identified.

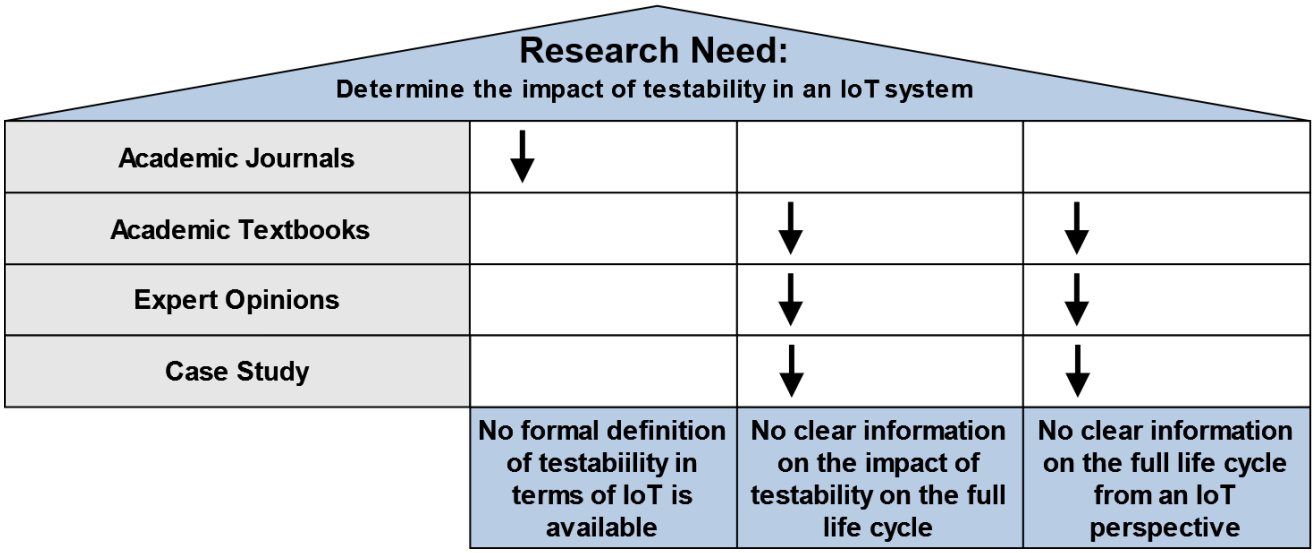

Figure 3.5: Completed Top Section of the RVM

3.4 Conclusion

The research problem was divided into focused research challenges to create a simplified approach to this research. Literature revealed there exists no formal definition for testability in the IoT context, which is critical for this study. Testability was found to be mostly limited to the scope of functional testing, and no clear information was found on what the impact of testability on the full life cycle. Since IoT is a relatively new concept, the information on the full life cycle from an IoT perspective was limited. The three research challenges form the basis of the study and stem from literature, industry observations, and a case study. As the RVM indicates, these challenges will be addressed individually, and a concept solution for each will be developed.

Chapter 4

Literature Review

This chapter addresses research challenges by linking relevant research topic areas to these challenges and by researching each area to identify existing knowledge, shed light on relevant challenges, and to assist in finding generic solutions that will address research challenges. The literature study section of the RVM is thus defined here, to form an essential link between problem and solution as part of the QRM method.

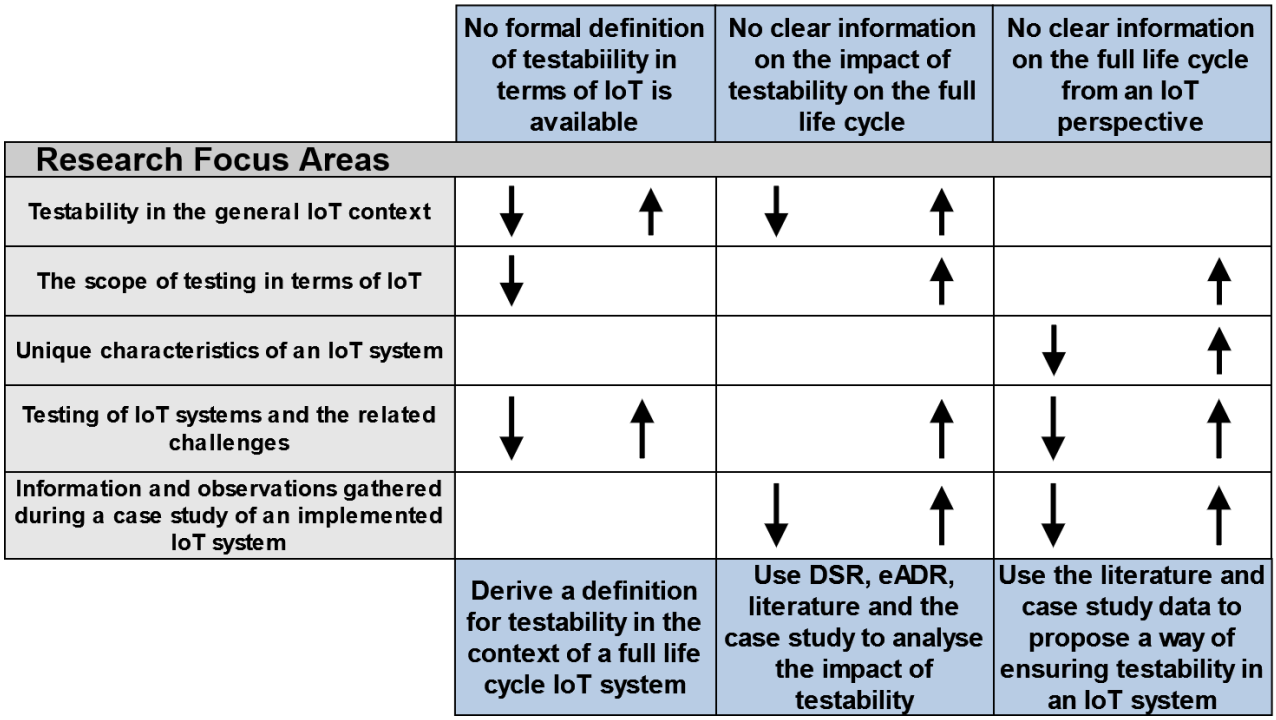

Figure 4.1: Completed Middle Section of the RVM

4.1 Systematic Literature Review

A systematic approach was taken to filter literature sources using set criteria to ensure the quality of these sources. This method not only ensured the academic sources are of good quality, but also reduced the amount of time spent searching for these texts [24],[25]. Inclusion and exclusion criteria were created, which were used to evaluate the texts before they were accepted in the study. The criteria were assigned trace codes to analyse and evaluate each source individually. Table 4.1 shows the inclusion and exclusion criteria used.

Three reputable databases were referenced: Researchgate, ScienceDirect, and IEEE Xplore. Key terms were used to link texts to research topics of this study. The terms include: "*IoT*", "*Testability*" and "*Testability in IoT*". All texts were required to either be an article, book, or journal.

Table 4.1: Criteria for Source Inclusion or Exclusion

	Criteria (Code)	Criteria Description
Inclusion	Closely Related (CR)	The text relates to general Testability or testing of IoT systems
	Application Relevance (AR)	The type of application used for IoT or Testability provides insight into the research problem
Exclusion	Duplication (DP)	The text appears multiple times within the same search criterion
	Language Compatibility (LP)	The full text is not accessible in English
	Text Length (TL)	The full text is not available or accessible
	Context (C)	The context wherein IoT or Testability is used differs too much from the context of the study
	Casual Use (CU)	The terms IoT and Testability are not discussed or investigated in depth to provide relevant information

4.1.1 Text Selection Process

Reference selection followed a filtering process comprising four distinct stages, with two exclusion stages and its exclusion criteria as indicated in figure 4.2. In the initial exclusion stage, references were discarded if the information was duplicated from another text (DP), not available in full English (LP), or the full text was not available (TL). After the first exclusion stage, references were removed in the second exclusion stage if the topic was found to be non-applicable (C), or the text used the terms testability and IoT in a casual way (CU) without providing in-depth definitions or discussions on the topic.

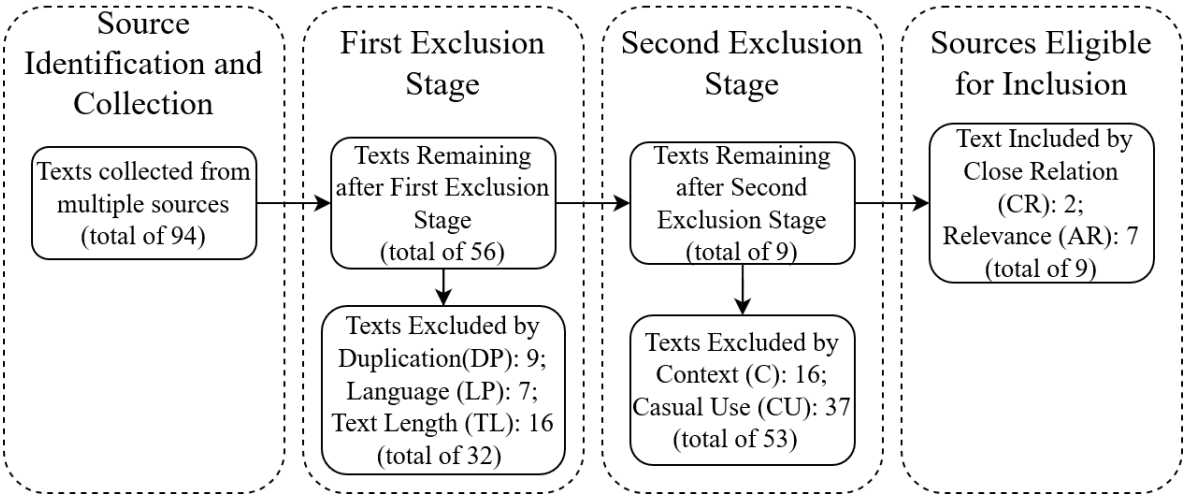

Figure 4.2: Four-Stage Systematic Text Selection Process, Adapted from [26]

The nine texts that were deemed eligible are captured in table 4.2, along with a brief description of the core ideas the text addresses.

Table 4.2: Core Ideas of Related Works

Title of Work	Core Idea
Internet of things (IoT) in the lab staging and testing for the real world [1]	General structure of modern IoT systems and the scope in which they are tested.
Internet of things: Current challenges in the quality assurance and testing methods [2]	Principle issues of IoT networks and their impact on testing methods.
Testing IoT Systems [3]	Investigation into the possibility of fully functional testing of IoT systems.
Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application & future challenges [14]	What defines IoT, what characteristics are unique to IoT systems, and how IoT can be applied.
Research Challenges for the Internet of Things: What Role Can OR Play? [15]	The challenges relating to IoT research and how they can be approached.
A Comprehensive View on Quality Characteristics of the IoT Solutions [16]	Classification of characteristics of IoT systems.
Testability [20]	Defining testability in a general context, how to identify testability, and creating a testability experiment.
Testability modeling usage in design-for-test and product lifecycle cost reduction [21]	Testability in system design, modeling for Design-For-Test, and uasbility.
Model-based testing for IoT systems: Methods and tools [22]	Testing of IoT systems by using abstraction and automation.

Four main themes were identified from these references, namely:

1. Testing of IoT Systems and the Related Challenges
2. The Scope of Testing in Terms of IoT
3. Unique Characteristics of an IoT System
4. Testability in the General IoT Context

The lack of clear definition of testability in the IoT context became evident as IoT testability focused mainly on IoT devices, as shown in figure 3.2. No literature was found on the impact of IoT from the perspective of a system full life cycle as shown in figure 3.3. In terms of testability, specifically in the IoT context and how it differs from conventional system testing, limited in-depth research was available. Regarding the focus areas, the following was found during the literature analysis:

1. **Testing of IoT Systems and the Related Challenges** [1],[2],[3]:
   The size of the IoT network has always been a significant challenge in the testing of IoT systems [1],[3]. This is because each node on such an IoT network is connected to multiple sub-systems and is required to handle communications, control signals, data capturing, data processing and reporting. The status of each node, as well as all the interconnections it is aware of, must be available at any time for an IoT system to be testable. In the literature that was reviewed there was no specific information considering the critical communication links in the network, but they must be considered in real-world systems for failure mode effect analysis.

2. **The Scope of Testing in Terms of IoT** [1],[3]:
   The scope of testing in the IoT context was found to be limited to mostly functional testing of devices and their individual features. Therefore, it was concluded that testability in IoT had to be redefined in the larger system context and that a clear definition of full life cycle testing in IoT systems was not fully researched.

3. **Unique Characteristics of an IoT System**[14],[16],[17],[18]:
   An IoT system is characterised as a large network of multiple different hardware, software, data, and human elements employed with the goal of generating, collecting, and processing data. Themes that were found to be common among all IoT networks were: interconnectivity; heterogeneity; dynamic nature; enormous scale; and data security.

4. **Testability in the General IoT Context** [20],[21]:
   The general context of testability is limited to test processes, procedures and methods in detail design, production testing, and diagnostic testing in the full system life cycle. From the literature review, no reference to full life cycle testing was found, thus identifying the need to analyse the system life cycle's high-level tasks individually. Such an analysis had to be done on both the system's acquisition and utilisation phases. In the general context, the definition of testability relies heavily on the concepts of controllability and observation testing as stated in [13],[19],[21],[26],[27].

From the above, the concept of testability had to be reduced to its simplest form. Testability can be described as the ability to establish the status of a system at any point in time and to compare that status against a desired reference state.

4.2 Life Cycle Phases and Research Focus

Analysis of the full life cycle revealed that the highest impact of testability, specifically in the context of IoT, is in the operations and maintenance tasks of the system utilisation phase. The main focus areas of testing in the acquisition phase was during development and production, which is well described in literature. However, the focus on operations and maintenance, again in the IoT context, was not evident from literature and thus allowed research to focus on these two areas.

Operations, as a high level task, uses IoT to provide production data which is also well defined in literature, and the only section that remained was thus maintenance, specifically "downtime". Interestingly, this was not foreseen when this research commenced and this information only became available after the literature study had been completed and reflection was done as part of the ADR process. Figure 4.3 defines the scope of analysis as obtained from a widely used text on SE [23].

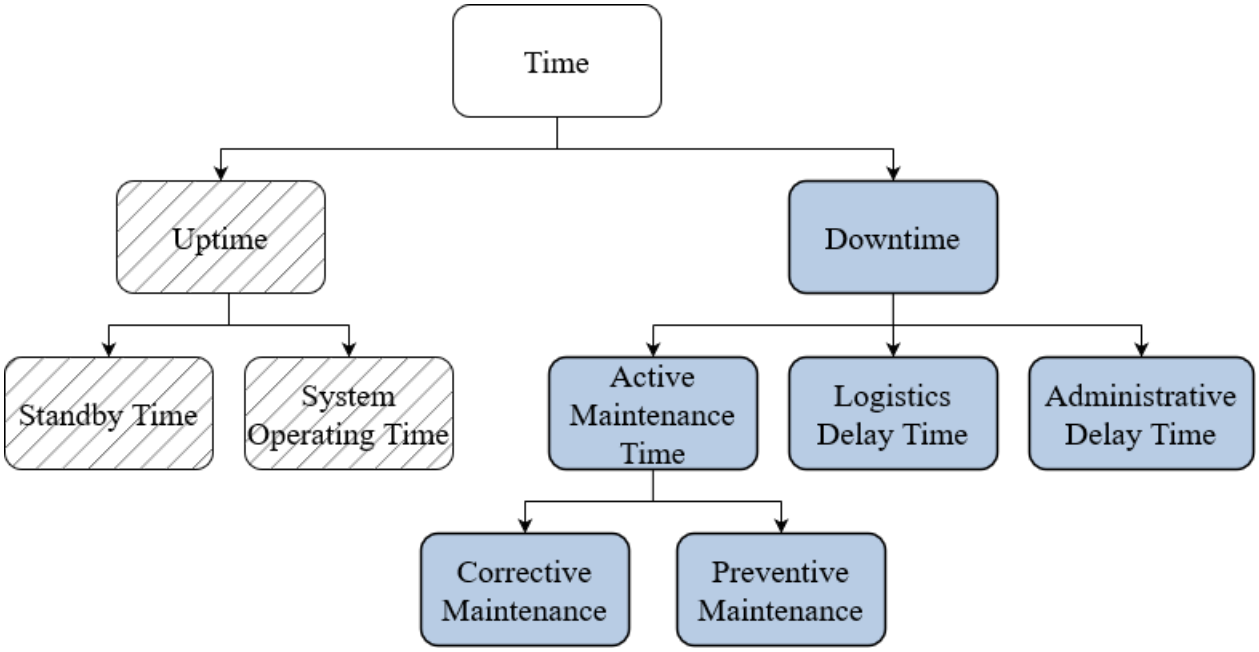

Figure 4.3: Composite View of the Study Scope of Analysis, Adapted from [23]

As shown in figure 4.3, the study focused on downtime and its elements as shown in the diagram. The investigation of the impact of IoT was focused on administrative delay, logistics delay, and active maintenance times. The active maintenance time was further broken down into corrective and preventive maintenance tasks that are discussed in section 4.3.

4.3 The Maintenance Cycles

One of the principle objectives of any SE process is to design and develop a product or a system that can be effectively and safely maintained with the minimum amount of time, cost, and resources [13]. Investigation into the testability of an IoT system conducted in this study revealed the impact of IoT on the maintenance task of the life cycle, which will consequently affect the overall testability. To understand in what way IoT will impact maintenance, the context of maintenance must first be defined. Maintenance, in this context, are actions to restore equipment or its functions to fulfil the role for which it had been designed [28].
Different maintenance paradigms namely, Time-Based Maintenance (TBM), Condition-Based Maintenance (CBM), Reliability-Centered Maintenance (RCM), and Run-to-Failure (RTF) have been identified in literature to describe the application of maintenance in different environments and circumstances [28],[29],[30],[31],[32]. From a systems engineering perspective, the different maintenance types can be classified into either corrective maintenance or preventive maintenance [23],[29].

4.3.1 Corrective Maintenance:

Corrective maintenance, or unscheduled maintenance, takes place as a result of failure, with the goal to restore a system to a specified level of performance [23]. Corrective maintenance is defined in eight stages, namely: (i) fault detection; (ii) fault isolation and localisation (diagnostics); (iii) disassembly; (iv) removal, replacement or (v) repair of faulty equipment; (vi) reassembly; (vii) adjustment; and (viii) system verification [23]. Figure 4.4 captures corrective maintenance as a flow diagram.

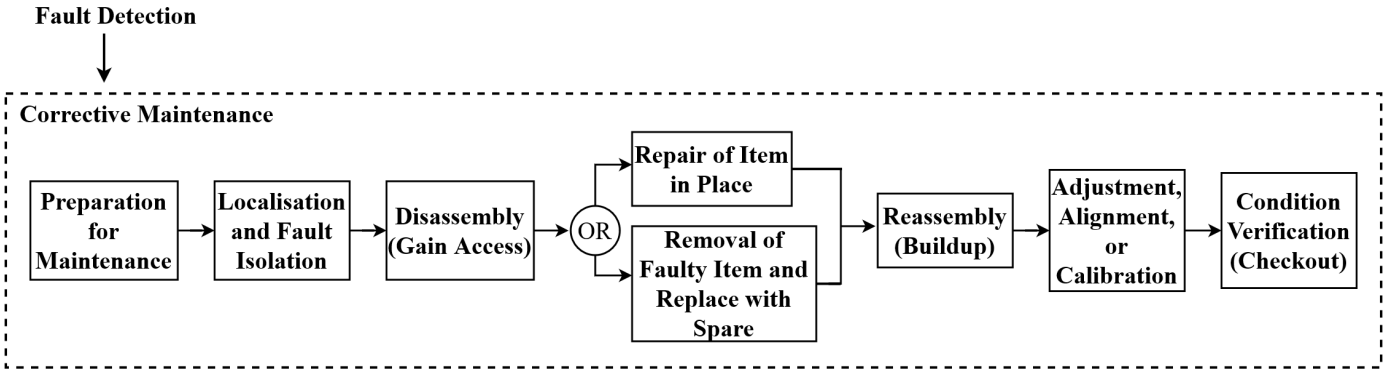

Figure 4.4: The Corrective Maintenance Cycle, from [23]

Most of the focus on the maintenance of systems is placed on preventive maintenance and its improvement. As a result, corrective maintenance has not been investigated in-depth in an IoT context [29],[30]. The only corrective maintenance being applied in most modern environments is the RTF approach and in some cases RCM [29].

- **Run-to-Failure** [28],[29],[30]:
  In RTF maintenance, equipment is allowed to run up to its failure point before it is replaced or repaired. This method of maintenance application allows the full use of the equipment, but continuously operating hardware towards the end of its Remaining Useful Life (RUL) will result in loss of production quality (in the case of factories or machines), loss in operational performance, or unwanted and unpredictable behaviour. For RTF to be a viable maintenance system, a large inventory of spare parts must be kept, requiring large areas for storage and a large initial cost. This maintenance method is feasible for non-complex hardware, as complex hardware failures may cause significant downtimes and high repair costs.

- **Reliability-Centered Maintenance** [29]:
  The main philosophy of RCM is to concentrate maintenance efforts on critical components of the system. The RCM follows a workflow that either flows to CBM, Scheduled Maintenance (SM), or RTF. It is done by monitoring a single defining variable that will describe the degradation of the component with defined threshold failure conditions. The first step in the RCM flowchart is whether component condition monitoring is possible. At this point, if an affirmative answer is given CBM can be considered a viable option.

4.3.2 Preventive Maintenance:

Preventive maintenance, or scheduled maintenance, is defined as an action that takes place to retain a system at a specific level of performance by providing systematic inspection, detection, servicing, or the prevention of impending failures through periodic component replacements [23]. Figure 4.5 captures the preventive maintenance cycle as a flow diagram.

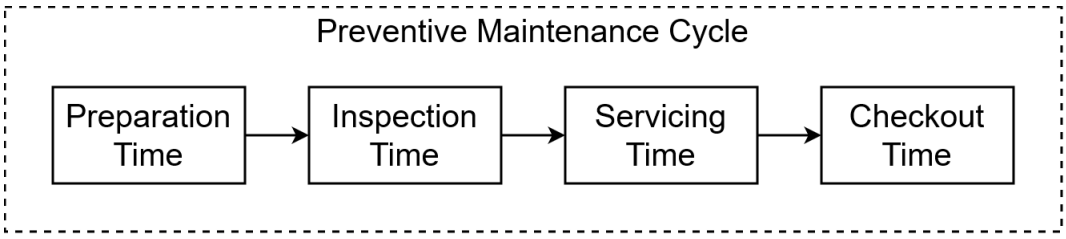

Figure 4.5: The Preventive Maintenance Cycle, from [23]

Preventive maintenance has been a focus area in research for quite some time [30] as preventive maintenance has large impact on the longevity of components, as well as reduced cost. Preventive maintenance can be divided into three categories namely, SM, CBM, and Predictive Maintenance (PdM) [29].

- **Scheduled Maintenance** [29]:
  This approach to maintenance is best described as maintenance activities on a fixed schedule. These simplistic maintenance activities include the basic replacement of consumable components, lubrication, and cleaning. SM on its own is a limited process, and the lack of component condition monitoring could cause unnecessary component replacement, causing an increase in downtime and costs.

- **Condition-Based Maintenance** [29],[30],[31]:
  CBM is a relatively new maintenance approach as a result of IoT. The application of CBM is described as a six-stage process with the following phases: (i) data acquisition, (ii) data manipulation, (iii) state detection, (iv) health assessment, (v) prognostics assessment, and (vi) advisory generation. CBM uses condition data to schedule maintenance actions when the degradation threshold of a component has been reached.

  Monitoring is done by human inspections, tests, and sensors in an IoT network. CBM resulted in PdM which uses the data captured by CBM to predict failure of components. These predictions reduce (and in some cases eliminate) the need for manual inspections and tests, decreasing the overall maintenance downtime and costs.

- **Predictive Maintenance** [29],[30],[32],[33]:
  PdM and IoT are complementary systems. PdM can be divided into (i) data capturing which is done in most cases employing CBM, (ii) data communication which is how the data captured is transmitted to the server, and (iii) the making of predictions regarding the RUL of the components based on the captured data. These predictions can reduce, and in some cases eliminate, manual inspections and premature maintenance which will decrease the overall downtime and maintenance costs.

4.3.3 IoT in Maintenance

Different maintenance paradigms enable advanced forms of maintenance [30]. IoT in maintenance focuses mainly on preventive maintenance such as PdM. With IoT and its complexity, advanced maintenance practises are becoming necessary [28]. A combination of multiple preventive and corrective maintenance paradigms is now used with IoT systems. As an example, complex IoT equipment health status and cost over time could look like the diagram in figure 4.6.

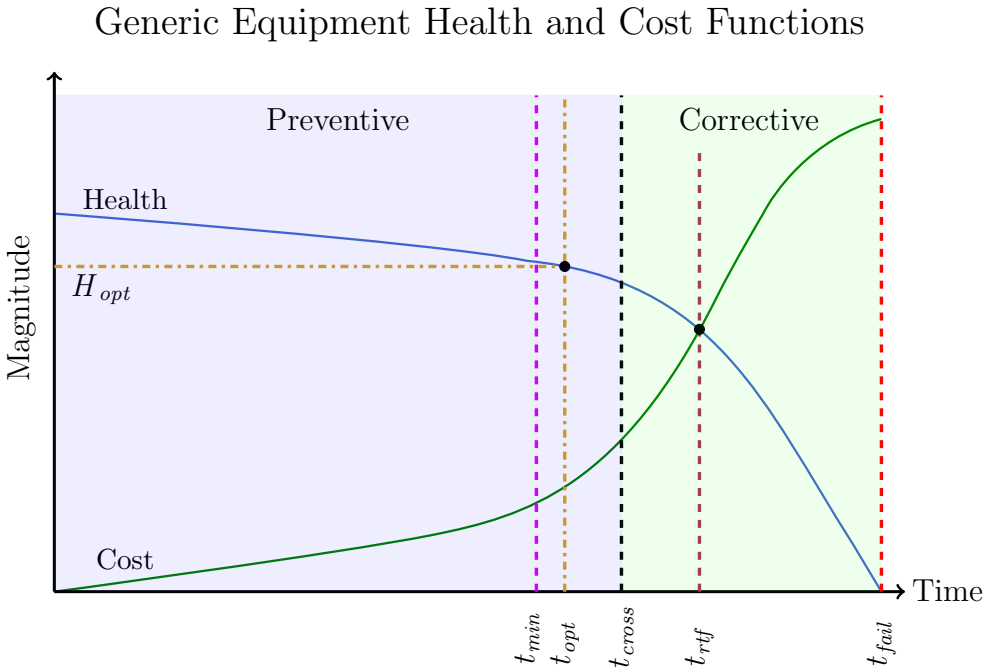

Figure 4.6: Relationship Between Equipment Health and Cost in Maintenance, Adapted from [28],[31],[32],[33]

The generalisation and estimation of equipment health and cost were derived from the multiple approaches to maintenance and how they are applied in different environments on different components and systems [28],[31],[32],[33]. In the figure, the blue line depicts the magnitude of the general overall health of the component or system which steadily degrades linearly, up to an estimated 70% of its RUL, at which point the degradation follows a steep downwards curve. The green line, which follows an S-curve, depicts the corresponding maintenance costs associated with the component relative to its health.

The graph above is divided into two sections by the t_{cross} line, namely preventive and corrective. The t_{cross} line indicates the point in time that the equipment health has degraded to a point where preventive maintenance is, in most cases, either not possible or is no longer economically feasible and a corrective maintenance approach must be strongly considered.

On the preventive side of the figure, three lines are evident: H_{opt}, t_{opt}, and t_{min}. The t_{min} line indicates the point at which the equipment manufacturer first recommends maintenance to take place, which could be premature when considering the overall health of the equipment [28]. The H_{opt} line indicates the optimal point where the equipment would require maintenance. The H_{opt} line is determined on a per component/system basis and is the product of a

combination of RCM and CBM. The t_{opt} line indicates the point in time the equipment health has degraded to the optimal health for maintenance. Note that the optimal maintenance point is located at a later point in time than the point where the manufacturer first recommends maintenance actions.

When considering the six-stage process of CBM, it is clear that IoT reduced human intervention while increasing the efficacy of the process. IoT addresses the largest shortcomings of CBM namely: (i) human error during sampling, (ii) large times between data sampling events, (iii) limited or difficult access to equipment for humans, and (iv) the high costs of sampling [32]. Data capturing, data manipulation, and state detection may be done by IoT-enabled sensors or devices with high data availability resulting from this automation. The processed data and equipment state would then be sent to a dedicated server to manage PdM where needed.

Predictive maintenance optimises the availability of equipment by detecting and diagnosing faults while predicting the RUL [29],[30]. Although the main focus of PdM is to provide reliable prediction of RUL, PdM also provides financial benefits. As a consequence, RUL predictions extend equipment lifetime and reduce maintenance costs and downtime.

IoT captured data from the CBM process is used in a near real-time manner to predict the H_{opt} point and activities to be scheduled accordingly. In addition, technology management can be applied to monitor technology performance across an estate, which will also improve availability and direct maintenance effort. Data from many remote sites with IoT-enabled equipment may be used to refine prediction models to also consider different operating conditions and environments. A refined model should be able to predict an optimal maintenance point with a higher degree of accuracy.

On the corrective side of figure 4.6, two lines are visible: t_{rtf}, and t_{fail}. The t_{fail} line indicates where equipment has reached its designed operational End-of-Life (EOL) and should, at this point, be replaced without maintenance consideration. The t_{rtf} line indicates the point where any equipment maintenance actions are deemed not economically feasible.

IoT in corrective maintenance is commonly overlooked as preventive maintenance presents a higher potential for decreasing operational and maintenance costs. With IoT, predictive maintenance can be applied in corrective maintenance. The focus of predictions may shift between phases from predicting an optimal maintenance point to predicting the RTF failure point of equipment. This can be used to decide if the equipment should be allowed to RTF, or to decrease the intervals at which SM should take place. The predictions can be used to assist in the decision-making of when to replace the equipment before the equipment reaches the point of failure and possibly cause damage to other pieces of equipment. The prediction of the failure point will also allow the ability to identify equipment near its functional EOL in order to source replacements early on, reducing the total downtime of the equipment.

To summarise, it is clear that IoT plays a significant role in maintenance. The exact impact of IoT has not become clear from the literature and a case study will be conducted later on to shed more light on this topic.

4.4 Digital Twins in IoT

The use of the term "Digital Twin" has increased in frequency over the last decade. The proposed use of digital twins in Industry 4.0 caused this. To better understand the concept and how a digital twin will be implemented in this study, we must first define what a digital twin is. A general definition most are familiar with was given by Chen [34], which described the digital twin as the following: "*A digital twin is a computerised model of a physical device or system that represents all functional features and links with the working elements*".

This definition would suffice in the general case, but a more specific description was required for this study. A more appropriate definition was given by Fuller [35], when he stated that a digital twin could be described as the effortless integration of data between a physical and virtual machine in either direction. This definition better describes the application in which a digital representation (twin) will be implemented in this study. With the concept of a digital twin being defined, common misconceptions exist about what exactly a digital twin is and how it fits into this context. These misconceptions can be explained using a digital model, digital shadow, and digital twin.

4.4.1 Digital Model

A digital model is what is most commonly thought of when the term digital twin is used. A digital model can be described as a digital version of a pre-existing or planned physical object [35]. The unique characteristic of digital models is that no automatic transfer between the physical and virtual models exists. This means that once a digital model of the system is created, any further change to the physical object will not affect the digital model [35]. The data flow of a digital model can be seen visually in figure 4.7.

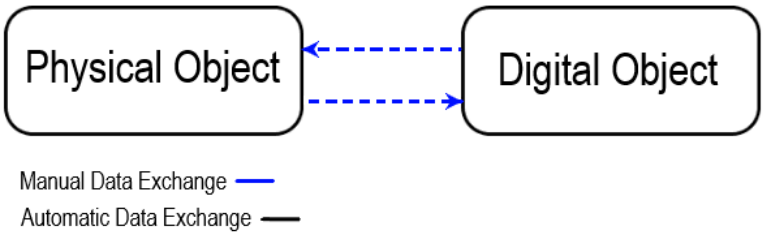

Figure 4.7: Data Exchange of a Digital Model, Adapted from [35]

4.4.2 Digital Shadow

A digital shadow is what can be called the mid-ground between a digital twin and a digital model. A digital shadow is described as a digital representation of an object with a unidirectional flow between the physical and digital object [35]. With a digital shadow, any applied change to the physical object will also change the digital object. However, with the data exchange being one-way, a change in the digital object will not change the physical object [36]. Figure 4.8 visually demonstrates the data exchange of a digital shadow.

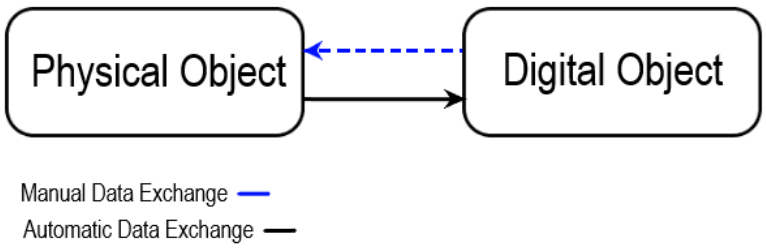

Figure 4.8: Data Exchange of a Digital Shadow, Adapted from [35]

4.4.3 Digital Twin

A digital twin can now be clearly explained in terms of the data flow. In a digital twin, the data is exchanged between an existing physical and digital object. The key difference is that the data exchange is entirely automated in both directions [35]. This means that a change in the physical object will bring about the same change in the digital object [36]. Similarly, a change in the digital object will bring about the same change in the physical object. Figure 4.9 below shows the data exchange in a digital twin.

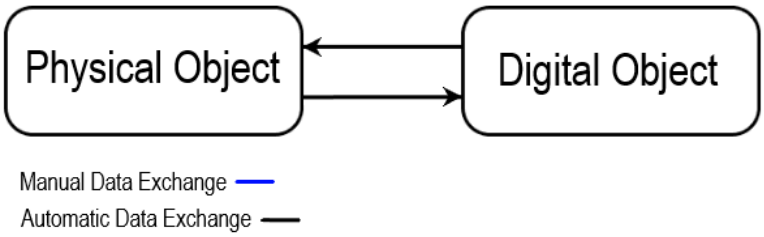

Figure 4.9: Data Exchange of a Digital Twin, Adapted from [35]

4.4.4 Digital Agent

The concept of a digital agent is not as widely known as it is not implemented as frequently. This could be attributed to the complexity of digital agent systems presently. How an agent can be defined is difficult as the application and considerations are unique to each case and context. Instead of a formal definition, we can define a digital agent according to a set of characteristics found to be frequently present in these systems, although it is not a requirement for the agent to have all the characteristics [19],[37]. These characteristics are listed as follows:

1. **Autonomy:**
   Autonomy is the characteristic of an agent to make decisions by itself without the need for intervention from either other agents or humans. According to the outcomes of the decision, the agent can also execute actions depending on the need, and control its own internal state [19].

2. **Flexibility:**
   A digital agent must be flexible in three different manners [19]:
   - **Reactive:** The agent is able to perceive its environment and react or adapt to any changes that occur over time.

- **Proactive:** The agent has a set of goals that drives its behaviour. This drive means that the agent is able to make goal-driven decisions and produce goal-driven actions by taking the required initiative to accomplish these goals.
- **Socially Connected:** The agent must be socially connected in the sense that it must be able to request assistance from other agents, humans, or other systems when it requires it. The social connection of the agent also allows it to share data, obtain data, and negotiate to achieve its goals.

3. **Situational Awareness:**
   The agent is fully aware of the environment in which it finds itself by the data it receives from the different sensors and devices connected to the network in the same environment. The agent must be able to use this data in a way as to perceive its environment and be able to modify its actions on all relevant levels to be fully situationally aware [19].

4. **Capability of Reasoning:**
   The agent can create knowledge from the information it has been provided with or collected. It can then decide, according to the information, which goal to pursue, suspend or abandon; which event to react to, or how to change its behaviour to accomplish a certain objective [19],[37].

5. **Adaptivity:**
   The agent is able to use its knowledge base and real-time captured data to modify its behaviour and actions and automatically adjust itself according to the information in the knowledge base or from the environment with which it interacts [19],[37].

The most common form of digital agent systems is what is known as a MAS. As the name suggests, it is implemented with multiple individual agents assigned to a small part of the network. These agents then communicate with each other to ensure the network functions as intended [38],[39],[40]. What complicates the MAS is the implementation of different levels of agents. The different levels, along with multiple agents per level, quickly increase the complexity of such a system and cause one to question the feasibility of such an approach. Captured in figure 4.10 is one of the most frequently used implementations of a MAS with the red dots indicating the agents on different levels.

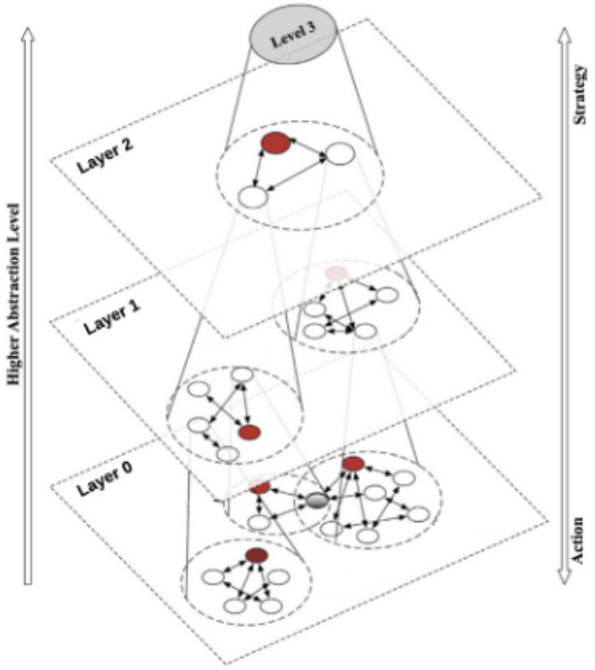

Figure 4.10: Holonic Implementation of a Multi-Agent System (MAS), from [38]

For this study, the implementation of the digital agent will be adjusted to fit our requirements. This includes, but is not limited to, reducing the digital agent system from multiple working agents to a single, high authority agent, as the concept of a digital agent is, in a sense, a digital supervisor for an IoT network. Such an agent would be a monitoring and reporting agent between the network admin and the IoT network. The agent would periodically monitor every node on the network and retrieve essential diagnostics from each node.

The digital agent would be implemented alongside a digital twin of the IoT network and be allowed access to each node's digital twins and the network as a whole. While the digital twins of individual nodes are being monitored, the agent also monitors the digital twin of the network as a whole and ensures all data connections are maintained and functional. The network admin then would only query the agent on the diagnostics and state of the IoT network and be provided with important information on all the nodes and the network as a whole. In figure 4.11, the role and implementation of a digital agent is visually depicted in a general IoT network.

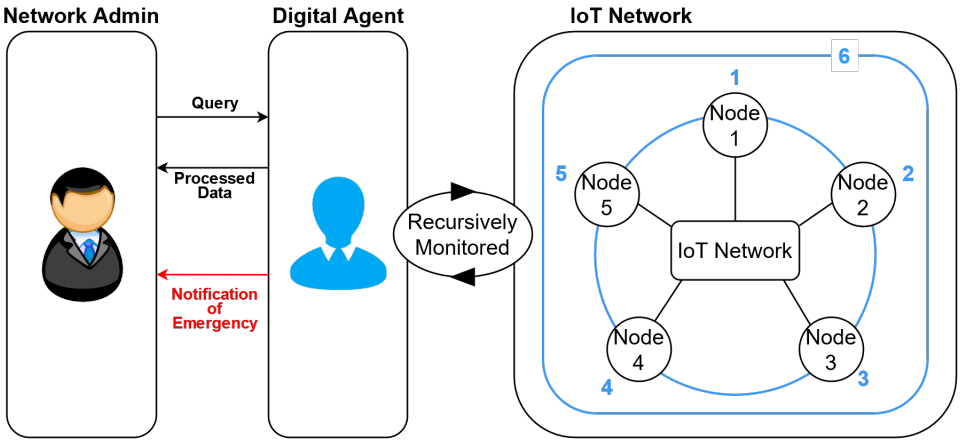

Figure 4.11: Role and Implementation of a Digital Agent in a General IoT Network

The application of such an agent could also improve the stability and reliability of any IoT network as any fault would be instantly detected and can be brought to the attention of the network admin without the need to monitor the network manually constantly. Implementing such an agent into any IoT network can increase the long-term testability of the network, as the agent will be able to provide data on the static and dynamic behaviour of the network and any nodes connected to it.

4.4.5 Multi-Layer Systems

A multi-layer system representation is important when considering both information and execution systems. Different architectures have been proposed with three, four and five-layer architectures being well defined in literature [19], the most common to use being the four-layer model [41]. Figure 4.12 shows the high-level four-layer architecture.

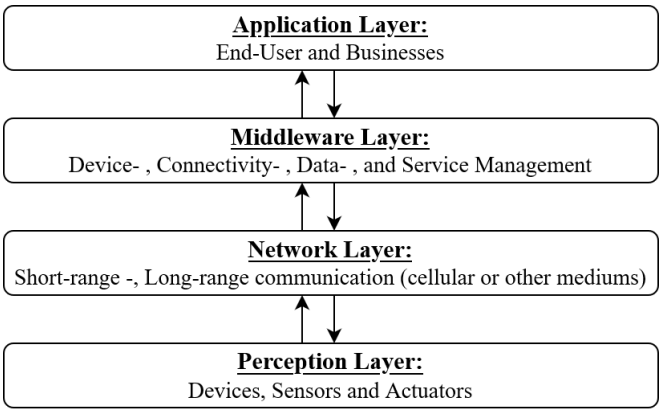

Figure 4.12: Four Layer High-Level IoT Architecture, Adapted from [19]

1. **Perception Layer:**

   The perception layer, or device layer, exists at the bottom of the IoT architecture. In this layer are all devices, sensors, actuators and any other real-world devices the IoT node or network is connected to [19],[42]. The perception layer's primary function is to capture data from devices in the real world. The nature of the data will depend on the environment inside which the IoT system exists and may include metrics such as: location, weather conditions, power consumption, water consumption, and device movement.

2. **Network Layer:**

   The network layer, or transmission layer, functions as a communication pathway between the perception layer and the middleware layer by transferring the information gathered by the perception layer to the middleware layer for processing [19],[42]. The principle function of the network layer is to connect devices to other devices and share information through the use of the internet [19].

3. **Middleware Layer:**

   The middleware layer, or service layer, is a cloud-based layer used for management. The middleware layer is responsible for the management and configuration of devices, connections, data and services of the system [19]. Decisions based on the data captured by the perception layer are made here and the necessary instructions are sent to the relevant devices or humans.

4. **Application Layer:**

   The application layer is the final and highest layer in the architecture. This layer is where interaction between the end-user and the IoT system occurs. The main focus of this layer is management of applications based on data received and processed by the middleware layer while supporting service requests from the end-user [19].

4.4.6 Synthesis from Literature - Definition of an Agent

The definitions of IoT, maintenance, multi-layered systems, and digital represenations above will be used in a case study later on to provide context and to show where IoT will have the most effect.

From a maintenance perspective, a case study will be analysed with the above findings in mind. The case study will be used to define areas where IoT adds the most value and to understand what a maintenance structure should look like.

The focus in this study will thus be on maintenance as part of the full life cycle since the other high-level tasks have been researched and applied extensively (as supported by observations from the literature).

It was seen that an agent is a supervisory entity that acts without direct human intervention to make the overall system status available (not only that of a device for which a twin is used). It is evident from the discussion that there is value to be added when an agent is used to perform data acquisition, processing, and reporting across all system elements as a whole, with the main focus on maintenance tasks.

4.5 DSR and eADR

A part of this study is the analysis of an already deployed IoT system in the form of a general field-equipment I/O controller of an irrigation Pivot system as a case study. A case study gives practical, real-world data to be evaluated along with relevant literature applied to the artefact creation part of the study. It will become evident how eADR and DSR are applied and how a digital representation of a system can be implemented to improve system testability.

4.5.1 DSR Application

As section 2.1.1 explains, the DSR paradigm creates a connection between the physical (real) and abstract worlds (knowledge base). The application of the DSR in the case study and the link it creates between the physical and abstract world is shown in figure 4.13.

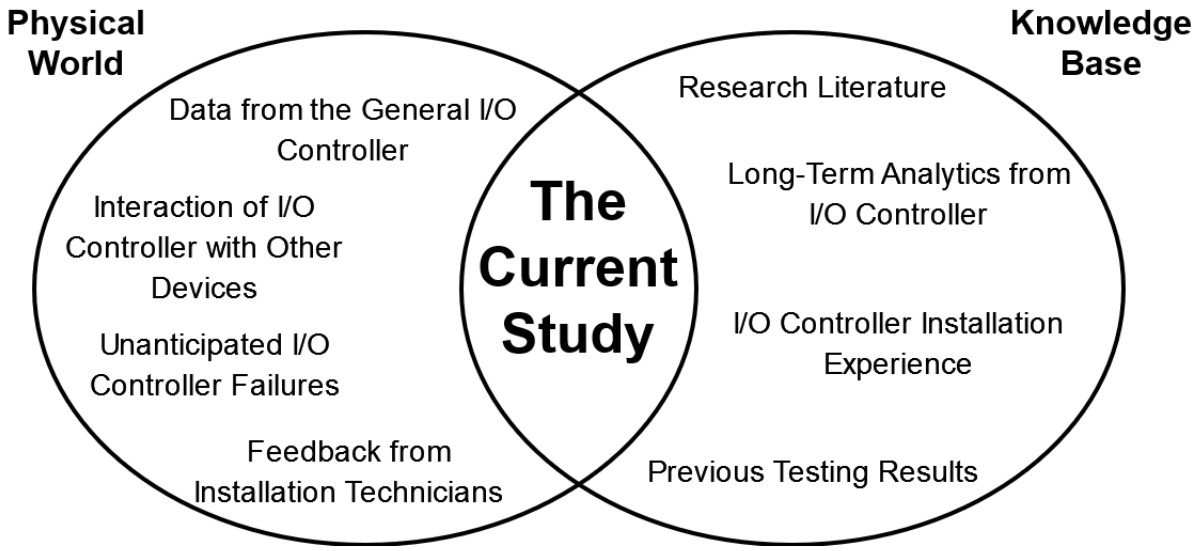

Figure 4.13: Application of DSR in the Context of the Case Study

4.5.2 eADR Application

Application of eADR, as seen from section 2.1.4, can be applied to systems that require design and development but is best suited for the analysis and development of a system after implementation to produce a relevant artefact as the product. Applying eADR in the context of the case study, the data collected can be analysed and employed, according to the framework in figure 4.14, to produce an artefact at the end of the study.

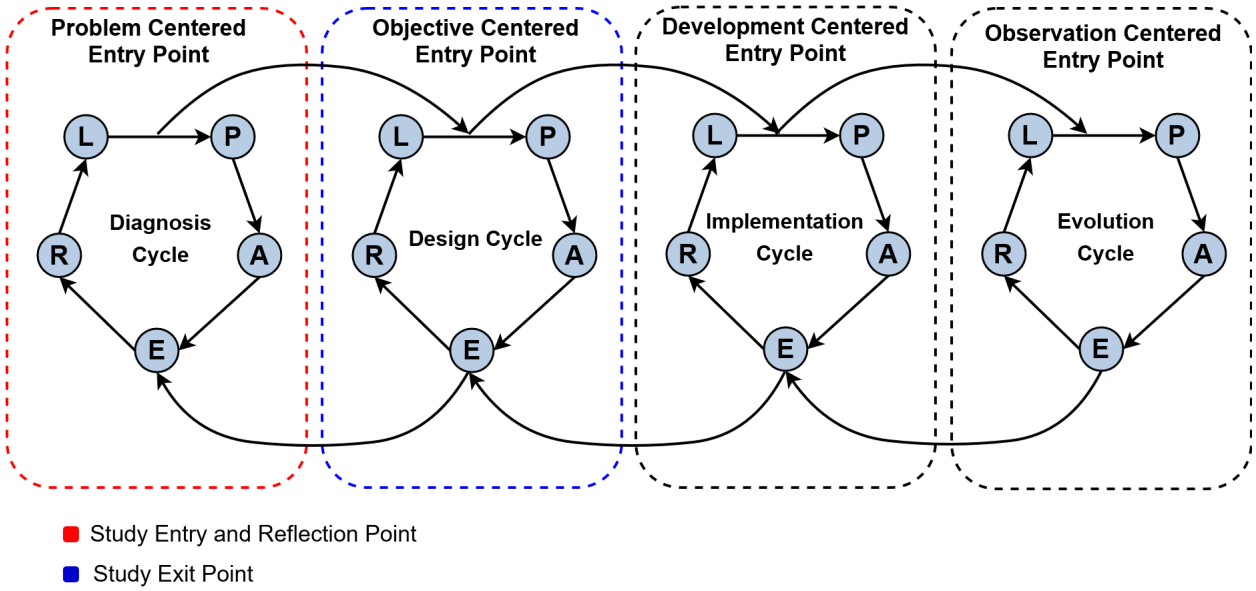

Figure 4.14: Application of eADR Chain in the Context of the Study, Adapted from [11]

The Pivot system used for the case study is, at the time of this study, in the operation and maintenance phase of the system life cycle. Figure 4.14 depicts the eADR cycles where reflection will occur during the development of the artefact. When an artefact is instantiated, it is physically implemented in the "Implementation Cycle", as opposed to being designed and proposed as a concept in the "Design Cycle" in the eADR process. In this research, the concept for an artefact is proposed as a result of the eADR Design Cycle.

The entry point of the study in the eADR process was thus the problem-centered Diagnosis Cycle since a clear understanding of the testability problem was not evident at the onset of the study. As a result, a clear view of a concept solution was also not evident, which indicated a need to start at the first of the eADR cycles. Analysis of the literature and the data collected from industry observations identified where and what the research problem was. The problem definition was revised in the diagnosis cycle of eADR. Revision was necessary as the knowledge base of DSR expanded and new information became available.

In this research, reflection and learning were done after each of the first two cycles, including the literature study findings and case study findings presented in the chapter on literature. These reflections were used to learn, define a concept, and verify the concept solution's applicability in the real-world case study.

4.5.3 Digital Twin Application

A digital twin, as explained in section 4.4, changes a physical object just as the physical object changes the digital twin. This relationship between the two objects allows the evaluation of a system in the static as well as the dynamic state. This immediately broadens the scope of testability of a system to not only the implementation phase where functionality is tested but also the evolution phase as described by eADR. A broader testability range provided by a digital twin allows the analysis of a system state in short-, medium-, and long-term. The system can be monitored and tested in the configuration phase of its life cycle, which can be categorised as a short-term analysis. Medium-term analysis can be applied to the time frame after deployment has been completed and the system has entered into the evolution cycle of eADR.

In medium-term analysis, static as well as dynamic analysis can be completed. Static analysis is possible when reconfiguration of the system must be completed in the case of failures or updates to the system. Dynamic analysis is possible when the system is functional and completing the task it was designed for. The digital agent can, in both states, monitor the digital twin and report to the main server or network admin to ensure the state of the IoT node is known at all times.

During long-term analysis, the system will mostly exist in a dynamic state. The analysis can be completed after the system has been deployed for a time, and no significant updates or changes will need to be made. At this point, the system is fully implemented and can be allowed to function until EOL. During long-term analysis, the digital agent will continue to monitor the digital twin of the IoT node and periodically report back to the network admin. The known state of a node on the IoT network is critical to the testability of the node and network, as testing is impossible without real-time data.

When considering the characteristics of a digital agent listed in section 4.4.4, it is important to note that all of these characteristics may not be present in an agent and what characteristics are present is dependant on the need. At the start of this study, the presence of agents in higher layers, such as the middleware and application layers, did not exist. The client system presented a need for a method to ensure testability is implemented, and a digital agent is a candidate option for implementing such a method. The client system makes both historical and real-time data available, making a proactive and reactive approach possible.

4.6 Testability in Context

4.6.1 Types of Testing

Implementation Testing

Implementation testing is what is most commonly thought of when testing of a system is being discussed. Testing during implementation, or functionality testing, is done in the implementation phase of the product life-cycle and only again when new hardware is commissioned. Elements tested during this type of testing include, but are not limited to, the wiring of the device, connectivity to the network (if applicable), and interfaces to the outside world.

In this phase, the application of IoT can reduce the number of faulty products, systems, or firmware releases that are eventually deployed. IoT would reduce the faults by capturing different test results of each unit being tested as well as essential details regarding the unit such as firmware version, hardware version, environmental conditions, and testing date. The unit will also be able to notify automatically of potential faults that may have occurred during the implementation phase when incorporated into an IoT network.

Observation Testing

Observation testing can easily be confused with calibration. Calibration is the act of comparing a measured value to a known value of high accuracy and adjusting accordingly. Calibration frequency is usually periodic or a once-off occurrence, depending on the device or product being calibrated. In contrast, observation testing is constantly monitoring current data or measurements and adjusting accordingly to produce the desired output.

An example of observational testing is a temperature-controlled greenhouse. The current temperature can be monitored, and the temperature control can be adjusted accordingly to ensure the desired temperature is maintained. The implementation of IoT can be very beneficial in this specific type of testing. When referring to the general I/O controller from the case study, IoT can allow for remote observational testing to take place. Since these I/O controllers can be deployed over a wide area, direct access to each controller can prove difficult and time-consuming. IoT allows these controllers to be monitored from a remote location and adjustments to be made in real time from these remote locations.

IoT incorporation enables the recording and capturing of analytic data that can present patterns in behaviour or environmental factors, which would not be available in the case of manual human observation. Such analytical data can provide important information that allows planning for future adjustments according to past behavioural patterns.

Installation of the general I/O controllers is simplified with the implementation of an IoT network. A physical installation occurs where the controller is deployed in the field, while simultaneously, all functionality and the controller's status can be verified remotely. This remote verification of functionality ensures that the deployment is successful and that no faults occur during field installation.

4.6.2 Testability in a Full Life Cycle

Testability can now be revisited in the full life cycle context across life cycle phases and high-level tasks as evident in SE [5]. Referring to figure 2.1, a full life cycle perspective requires a different view of testability in each phase. In the early life cycle phases, a testability design requirement is defined and applied to the following phases as part of system acquisition.

4.6.3 Testability During Development

In the development phase of the life cycle, a '*soft*' form of testability can be used. The product/system must be designed in such a way to allow for testability in the following phases. The most common way of implementing this is by means of a testability design model [21]. This model ensures the test considerations are made clear during the development phase with reiteration if the need arises. The model can be individualised further to specific sub-systems in the product/system as a whole, where it can be used to ensure testability in low-level design [21].

Testability design models also consider test accessibility. Test accessibility measures are defined in terms of controllability and observability [21],[43]. Observability measures are incorporated into the hardware and software design to allow the observation of the internal states at any point in time [21],[43]. Controllability measures are implemented to allow the establishment of a specific value in the product/system to drive the product/system into a known state [21],[43].

4.6.4 Testability During Production

During the production phase, both the major factors of test accessibility come into play. The product/system must be created in such a way as to allow both controllability and observability testing. For a software component, implementing the controllability factor means providing a form of 'testing' mode that the product/system can be set to that allows full access to all state parameters and inputs. For the observability factor, the same product/system must provide access to the current operational state of the system as well as access to all output data [21],[44].

In the case of a hardware component, the means of incorporating test accessibility is less clear. Incorporating controllability simply translates to a basic interface with which the product/system can be controlled. Similar to the controllability factor, the observability factor translates to a simple interface that can be used to collect data from the product/system [21],[43].

4.6.5 Testability During Installation

The installation phase testing is commonly thought of when using the term testing or testability, which usually combines controllability and observability. The controllability component is used to drive the software or hardware into a known state. The observability factor is used to observe the dynamic changes of the product/system to the changes brought on by the input of the controllability factor implementation [45],[46].

4.6.6 Testability During Operation

The operation phase is regarded as one of the final stages of the life-cycle [44]. The testing performed at this phase is final functional test procedures before a product/system is fully deployed and access is provided to the end user [21],[43],[44],[45]. These testing procedures take advantage of the controllability and observability test access designed into the product/system to verify the functional workings of the product/system with the original product functional requirements.

4.6.7 Testability During Maintenance

The maintenance phase is regarded as the final phase in the life-cycle [44]. Similar to the operation phase, the testing procedures performed on the product/system are limited and are restricted to functionality testing [45]. These testing procedures again take advantage of the controllability and observability test access designed into the product/system to verify the functionality of the product/system. In the case that functionality is not as expected, further action can be taken to correct the fault.

4.6.8 General Definition of Testability

Splitting testability over life cycle phases was necessary to fully understand testability. This, however, does not provide us with a clear general definition of what testability is in the context of IoT. It was clear that currently, the concept of testability relies heavily on the principle of test accessibility; specifically controllability and observability [21],[27],[43],[44],[45]. If a product/system has the ability to measure, evaluate, and initiate action, it can be considered as a system that has been designed for testability.

4.7 Synthesis from Literature Review

From the literature review, we can now re-evaluate and redefine the definition of testability in terms of IoT from a full life-cycle perspective. After evaluation, we can redefine testability as the ability to determine the status of a system at any point in time, to compare that status against set values, and to initiate action. We can use this base definition to expand and explore the impact of IoT and how this definition should be implemented.

The status of the system dictates the current electrical, electronic and mechanical status of all individual and collective components in a system. Access to data such as power levels, wired connections, and activated sub-systems for the electrical status is required in order to measure the status effectively. For the electronic status, measurements such as wireless connections, connection speeds and operational modes must be available. In the case of the mechanical status, metrics such as the current position of the system, the last service record and operational hours become important.

IoT in a system was seen to significantly impact corrective and preventive maintenance. The status of the system from IoT functionality has broadened the scope of what is possible in terms of employable maintenance. The widened scope is critical when analysing real-world systems/data as the economic impact in terms of downtime of the system from maintenance remains a large focus area for system improvement. Literature revealed the possible benefit that IoT may provide in such a system by means of early fault detection, fault isolation, and efficient scheduling of maintenance tasks according to CBM or PdM.

Testing in a system refers to measuring a parameter and comparing that parameter to a pre-set value or standard that dictates what value a specific parameter should ideally be. These pre-set values or standards differ depending on the parameter that is being tested and can range from internationally accepted standards to values pre-defined during the early design phase of a system.

The testability of the design must be kept in mind from its inception to allow for autonomous reporting of the system status. During the final stages of the development stage, the IoT system must be tested for functional capability to ensure that the IoT component/system itself is testable. When the usability of a system is addressed, testing is simple, elegant, and intuitive. Usability in a system supports testability.

System components will be tested by means of a component's system digital twin, and the results may be reported back to operators by means of a higher-authority digital agent. The digital twin of the system can measure relevant parameters and compare measurements against standards and pre-set values. In the case that a test has failed, relevant parties can be notified or, if the system was designed in such a way, the digital agent can rectify the error and record such a failed test.

To summarise, the definition of testability of an IoT system from a full life-cycle perspective reduces down to the ability to determine the status of a system at any point in time, to compare against desired values, and to initiate action. Digital twins and an agent may be implemented to further monitor the system status and report results to relevant resources for action.

Chapter 5

Case Study

A case study, used in research as a data collection or validation method, is an accepted way of explaining real-world concepts and occurrences. The case study in this research includes different perspectives, namely: exploratory, descriptive, and explanatory, as defined by Runeson [47]. The study focuses on determining the impact of IoT on a system's testability, where the impact on the life cycle is most evident, and how the life cycle was impacted in the real world. This chapter completes the final part of the RVM as shown in figure 5.1.

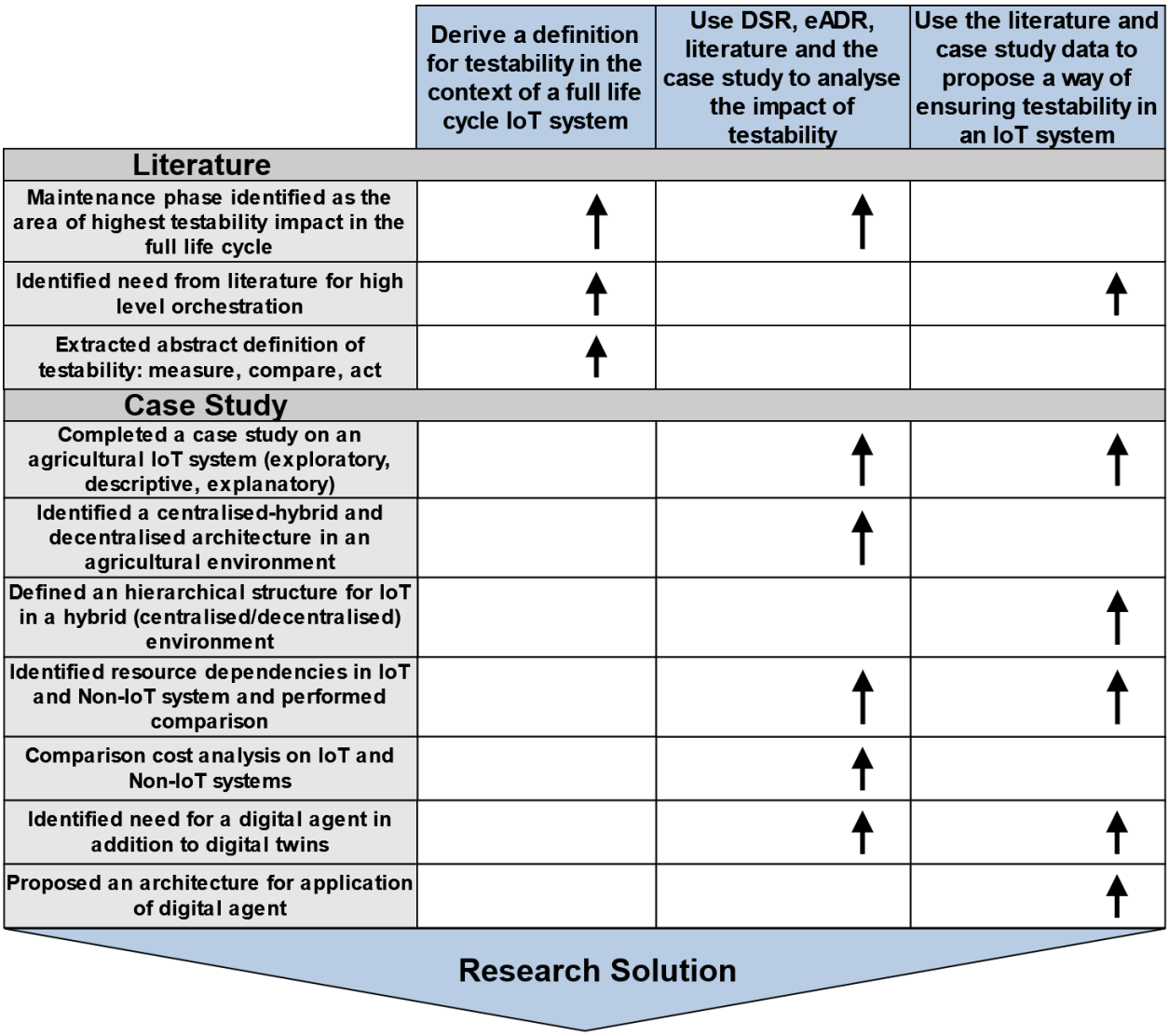

Figure 5.1: Completed Bottom Section of the RVM

5.1 Case Study Design

The objectives of this case study are identified along with the exact case, units that will be analysed, and the context inside which the analysis will take place. Finally, the research questions to be answered with the case study are presented.

The case study is intended firstly as an exploratory study with elements of a descriptive and explanatory study [13],[47]. The study focuses on determining the the impact of IoT capabilities on the overall testability of the system. Specifically, the study will investigate where the impact is the highest in the system life cycle, how it presented itself, and to what extent the overall system testability was impacted. Significant focus was placed on the utilisation phase of the system with special attention to the support of the system in general and the scheduled and unscheduled maintenance cycles.

The case study was used to analyse the effects of IoT capabilities on test efforts and cost in the operation and support tasks. Repair of a system may take place remotely (software changes, system reconfiguration) or on-site (hardware changes, peripheral system malfunction). The units of analysis used in this study are the individual work effort (time) and associated costs (a qualitative view on resource dependency). For example, travel and repair time and costs associated with a site visit due to absence (or presence) of decision support data. It was anticipated that an outcome of effective testing would result in a reduction of total system downtime and testing costs.

5.1.1 Case Study Participants

The case study included multiple parties, each with their respective role as listed below:

- **Primary Researcher:** The author of this document, taking on the role of both observer and participant by doing both pre-installation and installation testing, testing as part of the corrective maintenance cycle, and recording and analysis of the observations.
- **Research Leader:** The research leader assisted in the construction of the research, case study, and analysis of data. In addition, the leader assisted with the reflection and generalisation of findings.
- **Technical Team:** The IoT company's team responsible for on-site installations and testing and monitoring of system status during operation (for maintenance that is done by the farmer).

- **Cloud Development Team:** The IoT company's team responsible for development of software in the middleware layer of the IoT architecture as shown in figure 4.12.
- **End-User Team:** The farmer or foreman, responsible for performing inspections, non-specialist on-site repairs, and all service tasks.

5.1.2 Case Study System Context

An agricultural irrigation Pivot point and its sub-systems are shown in figure 5.2. This system consists of multiple IoT nodes in the form of individual measurement and control points. A single local master node acts as the GSM / cellular gateway.

In the case study, the different nodes include weather, electrical, and water supply points, interconnected wirelessly to the IoT master by means of a local IoT network. These nodes provide specific information regarding the performance and status of each sub-system to an I/O controller (Pivot controller) that is, in turn, linked to a central control centre.

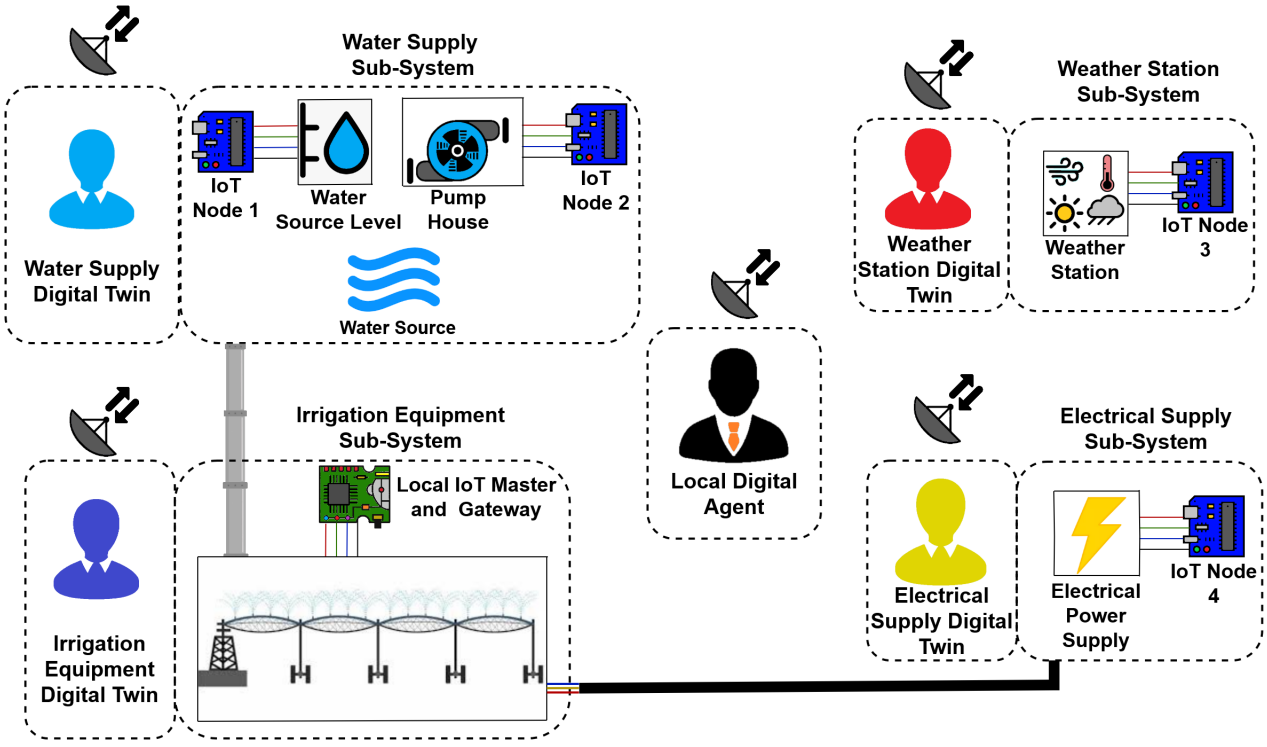

Figure 5.2: Irrigation Pivot IoT Physical Configuration and Interconnectivity

5.1.3 Rationale

This study provides a practical perspective in addition to the theoretical literature study. The academic study revisited the definition and implementation of testability in the IoT system context. As the practical implementation of IoT is relatively recent in the form of Industry 4.0, it is necessary to understand the physical elements of testability in IoT.

5.1.4 Objectives

This specific case study was conducted on an existing IoT system with around 300 individual Pivot field systems distributed across different remote sites. These Pivot field systems had different owners/end-users and included the Pivot as well as all the supporting sub-systems. Studying an already deployed system makes it sensible to identify testability in a full life cycle (as the operations phase has been entered). The IoT system case study started at the installation of a Pivot system. It ended in the maintenance phase, which limited the scope of this study to the utilisation life cycle phase.

5.1.5 Cases and Units of Analysis

An IoT company provided access to their very large and stable IoT Pivot system estate, including access to a control centre. The Pivot system was used as both the case and the unit of analysis in the study. The IoT company utilises an established, well-functioning IoT managed cellular network. Pivot systems are installed in remote locations and are provided to farmers with access to a wealth of IoT data that the farmers may use for maintenance purposes (the farmer is responsible for all maintenance, excluding IoT system maintenance). Testability of the complete Pivot system is considered a critically important characteristic for both the manufacturer and the end-user. An overview can be seen as figure 5.3.

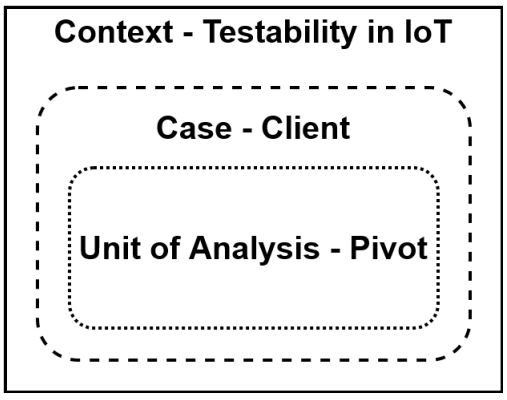

Figure 5.3: Overview of the Context, Case and Unit of Analysis

5.1.6 Research Questions

This study provided supporting evidence in a more extensive study. The question in this study is as follows: "What would the effect be on testability of the system with and without IoT?". The answer to this question provides valuable insight into how the implementation of IoT affects testability in a larger system context.

5.2 Planning

The methods of data collection and data selection are discussed below, followed by the study protocol and ethical considerations. Finally, the explanatory and descriptive goals of the study are given.

5.2.1 Method of Data Collection

The primary source of information was first-hand installations and operation of the Pivot systems from observed behaviour along with supporting measurements. The data required was more implicit and was not directly measured or calculated. Multiple field systems were included in the study (300) and observed to ensure a large sample size was used in the study.

5.2.2 Selection of Data

A factor to account for was multiplicity, and the variety of analysed units necessitated a fairly large sample size. To ensure a wider variety of samples, units installed in different environments and in different environmental conditions were used for the data rather than multiple units deployed in similar environments.

5.2.3 Case Selection Strategy

The IoT company provided a well-established, proven IoT network. It also provided study-relevant information in a timely manner and access to Subject Matter Experts (SMEs) that were part of the development and installation teams. The case was selected as the Pivot system had been deployed for around 4 years and has undergone field testing.

5.2.4 Case Study Protocol

Since the data collection was done at multiple sites in different environments and conditions, a protocol was needed to ensure consistent and reliable observational data was captured. The protocol was developed for the installation and operation of Pivot systems, both instances where observations were made. The implementation of this protocol ensured repeatability.

Installation Preparation

Systems to be installed system had to be fully pre-tested for all functionality, including the IoT connections. In short, all form, fit and function elements had to be verified before installation.

Pre-Installation Inspection

On arrival at an installation site, multiple inspections were done before an installation could commence. These inspections ensured the installation could proceed safely and effectively.

General Inspection: An overview inspection was done of the environment to determine if the environment was free from hazards that could affect the installation. A site was also photographed to document its initial state, including any already installed equipment before installation occurred.

Pivot Center Inspection: The inspection of the Pivot system's structure is highly technical. During the structural inspection, the inside of the currently installed control unit was photographed, along with the technical settings and the wiring connections. This was done to provide a reference to the installation technician(s) later on when a new control unit was installed. The reference included essential information regarding the Pivot, such as wire connections and motor phase placements.

Installation

During installation, critical checkpoints ensured a successful installation would take place. The first was to disconnect a currently installed control unit. The disconnection procedure included the disconnection of the 3-Phase AC cables, motor cables, control wiring of the Pivot, and any other additional sensors and wires.

Following a control unit's removal, a new unit's placement was determined. The new controller placement was heavily dependent on the length of the step-down power transformer cables that provide power to the Pivot, after which a new control unit was mounted.

Lastly, wire connections were made to the new control unit. The control cable harness, 3-phase AC, and any additional sensors or wires were connected. The installation of the control wires followed a specific standard wire colour scheme which could be used as a reference.

Post-Installation Inspection

After the wiring connections of the new control unit had been completed, a post-installation inspection was done. This inspection was done to provide evidence of the completed installation and to verify that everything had been completed to an acceptable quality and standard. The unit was photographed and checked. Finally, the site was photographed to provide evidence of the state of the site after installation.

Testing and Calibration

After the installation and post-installation inspection, a new control unit was field tested and calibrated. If all required tests had been passed, the technician(s) calibrated the new control unit to ensure accurate site-specific measurements were possible after commissioning.

5.2.5 Data Validity

The Pivot system is a fully tested and deployed system. As is standard in any IoT system, ongoing software and hardware upgrades are always done. As a single hardware version was used throughout the estate and software was updated uniformly, there were no differences between Pivot systems in the field, and the only variability was due to environmental differences.

5.2.6 Ethical Considerations

No individual or personal data was used in the study. Hence the study was classified as a low-ethics risk study. The data captured regarding the Pivot system and the client as a company was protected by a non-disclosure agreement between the IoT company, the researcher, and the research leader to protect all personal information.

5.2.7 Exploratory Study

The case study was structured as an exploratory study to generate knowledge from participation and observation in the problem analysis phase of the eADR process [47]. Observations were captured from two perspectives, namely descriptive and explanatory. Descriptive analysis was done to show what an existing system would provide in terms of testability using IoT, while the explanatory component addressed cause-effect relationships (for example, linking abstract architecture to observed centralised architecture).

5.2.8 Descriptive Study

The descriptive element defined the current system status for installation, operation, and maintenance. These observations were used to compare system testability with and without IoT. Thus, observations made from the descriptive analysis defined the relationship between the Pivot system and IoT, and how the installation, operation, and maintenance benefited from its use.

5.2.9 Explanatory Study

The explanatory study element linked observations to the relationships between layers of the IoT system architecture. The explanatory perspective linked observed architecture to abstract architecture, which resulted in a centralised view with a digital agent as coordinator - that is, a cause-effect relationship could be established.

5.3 Data Collection

Case study data was captured in the form of observations. Only observations relating to testing of the system in each system task, such as installation, maintenance and repair testing, were recorded.

By utilising the IoT capabilities of the Pivot system, the system data management process was analysed. Indirectly, observations were made on the effectiveness of measured data, including metrics such as: water supply; electrical supply quality; fluctuations and failures; network connectivity; and Pivot operational status. This data was not recorded as part of the study, but the process was observed and analysed in order to understand the overall IoT system (as opposed to detailed data analyses, which did not form part of this study).

As part of the observation, operations data was collected to characterise and describe the effects of the interactions between humans and the system on site. The operations data also reported measurements relating to the performance and functionality of individual sub-systems.

Observational data was gathered by individually analysing data from 300 Pivot systems in the field. Observations were used to identify points of failure, available measurement data, and actions taken to address failures.

5.3.1 Installation, Operation, and Maintenance without IoT

From the analysis of the system, a high-level diagram was derived as shown as figure 5.4.

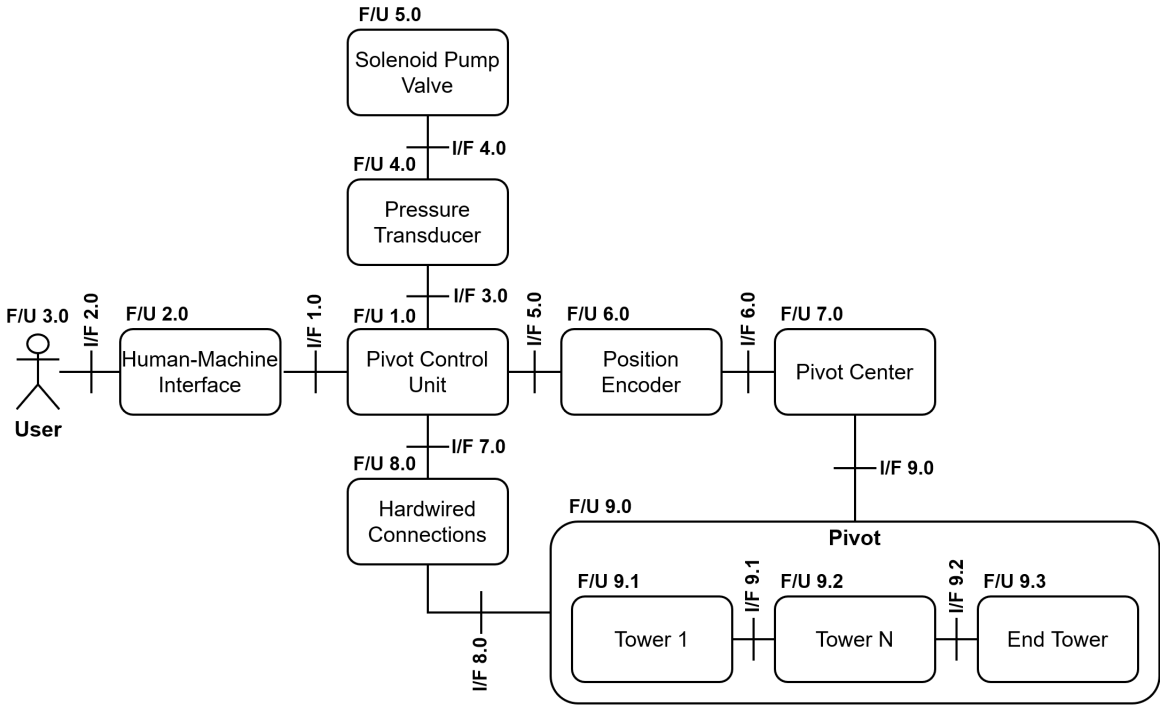

Figure 5.4: Pivot System Without IoT Integration

From figure 5.4, it is clear to see that, at a high level, the Pivot system consists of multiple individual components. Each component must be integrated and tested when a new control unit is installed, which is time-consuming. During testing without the use of IoT functionality, time and effort are spent waiting for a test to be completed. A run test, for example, must be done by allowing the Pivot to move in the desired direction for a set amount of time to monitor if a safety trip will be triggered. During this time, the technician(s) must wait at the testing site, as opposed to spending time installing new units.

If a Pivot does not pass all the required tests and fault finding must occur, the lack of IoT functionality dramatically increases the amount of time required to find the fault. The problem faced is that the state of only a single element of the Pivot system as a whole can be seen or inspected by the technician(s) at a time.

In maintenance, the system's lack of IoT requires frequent system monitoring by either the owner, supplier, or a designated maintenance company. In the case of the owner, this often requires unnecessary and time-consuming trips to each Pivot to ensure proper functionality. In the case of the supplier or designated maintenance company, this would require regular dispatching of a technician(s) to monitor the Pivot systems' state, resulting in lost person-hours on other projects. Even in cases where, for example, SMS messages are used, the lack of information content (and context) would limit decision-making effectiveness, and site visits are still required when failures or adjustments occur.

5.3.2 Installation, Operation, and Maintenance With IoT

To visualise where IoT functionality fits into the system, the IoT system is shown in figure 5.5 below.

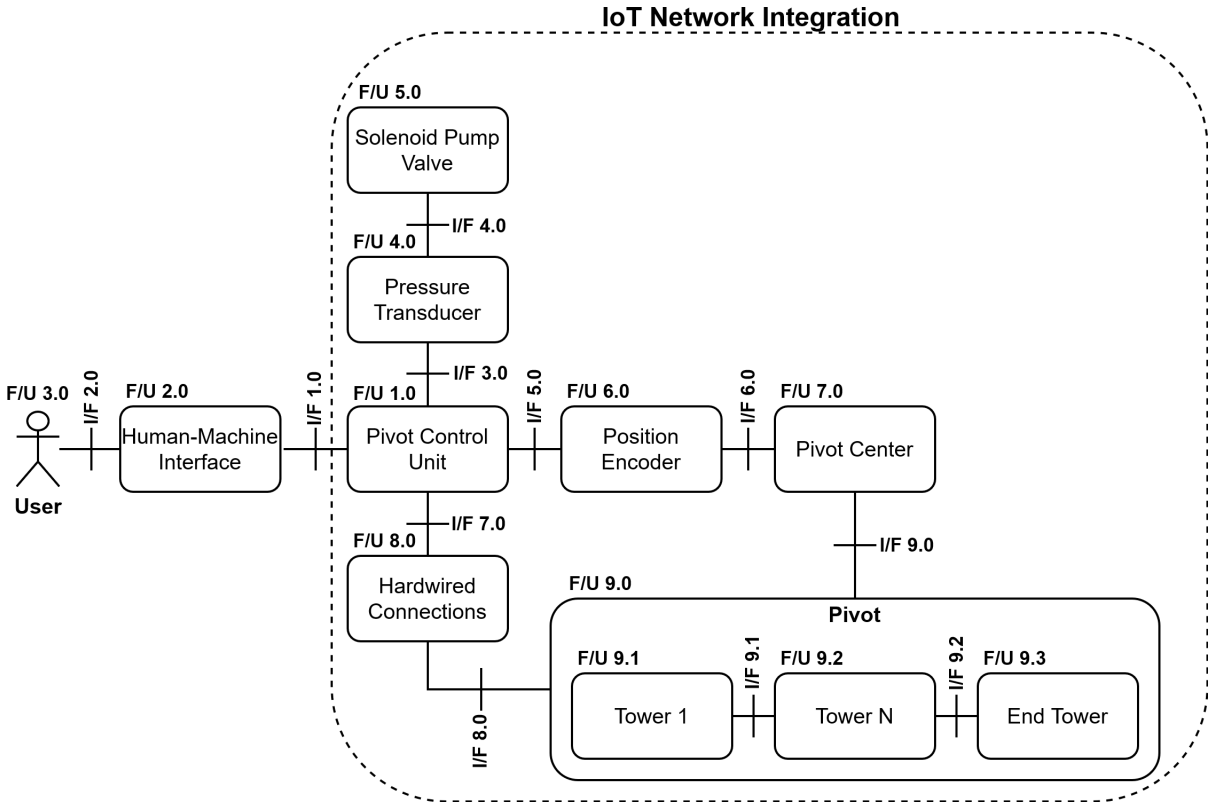

Figure 5.5: Pivot System With IoT Integration

The main advantage of a system with IoT functionality is effective testing, simplified fault-finding and directed repair activities. The Pivot system can be tested and left in a running state while the technician(s) move on to the next installation. A remote operator can monitor the Pivot and be notified instantly in the case of a fault during the test. If no fault has occurred, the remote operator can set the Pivot to the stopped state, or the technician(s) can be notified and return to correct the fault.

With fault-finding, a remote, experienced operator (centralised SME) is in contact with technician(s). The visual advantage that IoT provides is to make faults visible at a central location as well as on mobile devices. A technician(s) can only see a single system element at a time, but a remote operator can monitor systems across an estate. The remote operator can directly identify system faults and instruct technician(s) to focus efforts on relevant system elements.

IoT removes the need for individual monitoring by any one person as the system remotely notifies all relevant parties in case of a malfunction. Upon detection, the IoT company assists with the identification of the malfunction and assists with repairs, depending on the nature of a fault. Technician(s) can prepare appropriately and have access to current and historical fault data. From a testability perspective, it is evident that testability plays a critical role in system maintenance.

IoT allows the farmer to control and monitor the Pivot system remotely. This is especially important in cases where the farming area is expansive and logistically challenged. Access to system status data simplifies and streamlines on-site efforts significantly.

5.4 High-Level Orchestration Layer

From the study, it was seen that different sources of information were used to obtain the "system" status, including observed data (human), measured data (system), environmental and historical site data. These have to be combined to provide a single picture of "system status" and to compare that status against set points in the spirit of testing. The system, in this case, is thus the total system of process, people, technology, and data and not only the IoT or Pivot system in isolation. Hence, the need for an intermediate layer supporting data collection and presentation was identified.

The case study thus identified the need for an intermediate layer in the network that allows seamless IoT cloud integration. This layer, which will be referred to as the orchestration layer, is commonly implemented in digital form.

The IoT orchestration layer, in our context, is used as an umbrella term for the supervisory and management layers that make it possible for an automation component to be included alongside IoT in the larger system while also including humans in a supervised manner.

Figure 5.6 depicts a three-layer hierarchy, with each layer's authority level and the entities captured in each layer. The high-level orchestration layers, in our case, are used to describe different layers of humans, machines, equipment, and digital agents. The orchestration layer is thus depicted as more than an "IT automation" structure as orchestration tasks can also be executed by humans.

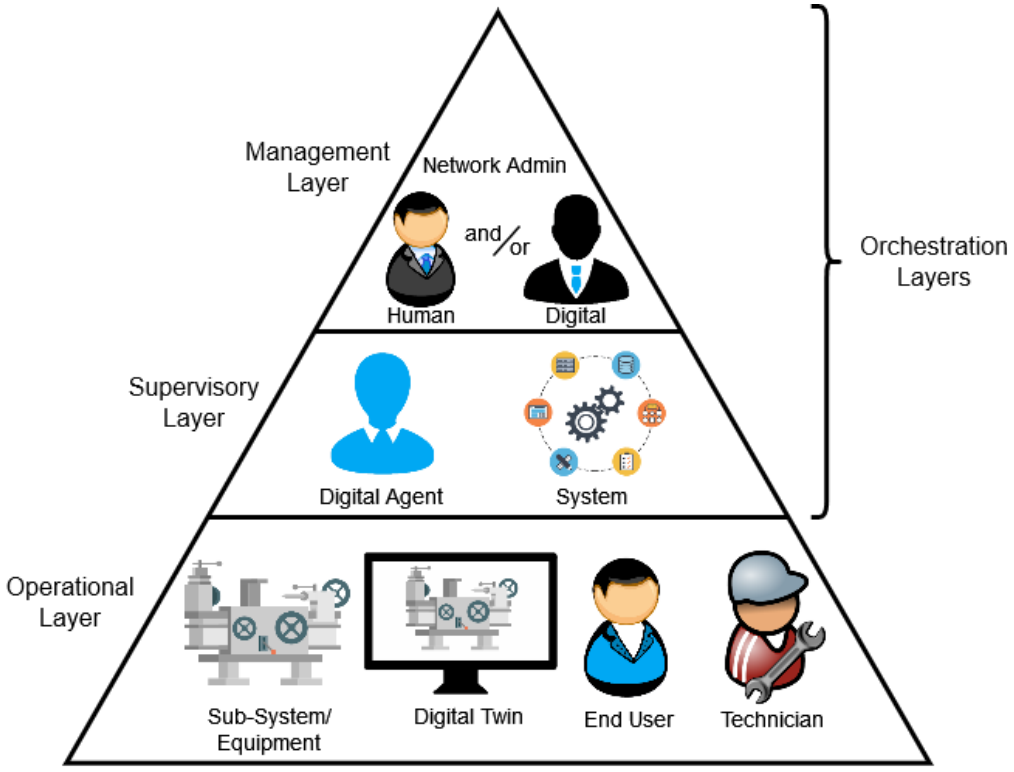

Figure 5.6: Hierarchy as from the Case Study

The high-level orchestration layers of a system allow system automation by utilising humans and technology in a structured manner. For example, as shown in the figure above, repetitive events and activities are assigned to the digital agents, with the ability to inform humans in a supervisory position (or intelligent higher-up systems) according to predetermined business rules.

When considering the IoT context, it is clear that testability will remain only a concept of design when the automation component is removed from the system. This is caused by the significant amount of recurring status monitoring, data processing, and reporting that the system requires.

When considering why testing should take place, it must be understood that the orchestration layer lives on IoT and associated system status data. This data is generated constantly from digital twins and agents that conduct status queries and measurements, report functionality and performance, and request human intervention. Without an IoT network, status data, and system metadata, the orchestration layer cannot function, and the bridge between the physical world and the IoT cloud will not exist.

5.4.1 Operational Layer

Operational entities, including people, machines, equipment, tools, facilities, and data, share the same level in the system. These entities provide status data through respective interactions, as well as through interactions with the system. Status data is obtained from direct functional performance and diagnostics of the physical system and indirect data derived from interaction with connected elements. For example, as observed in the case study, keystrokes and sequences from an operation inform the digital agent on the skills levels and failure modes of an operator in the field. The sub-systems present in this layer specific to the case study include the following:

- **Electrical sub-system** - Eskom-provisioned electricity status, as monitored by IoT equipment for availability and quality (directly measured);
- **Water sub-system** - Status of water pressure, flow, and availability, also monitored by the IoT equipment on site (directly measured);
- **Operator activity** - Status data of all operator(s) by monitoring user activity on IoT and connected equipment (directly and indirectly measured);
- **Maintenance logistical sub-system** - Site visits and site activities that can be measured on-site, as well as repair data from manual records (indirectly measured);
- **Environmental sub-system** - Weather / environmental status, as well as IoT equipment temperature and security data (directly measured);

- **Operational equipment** - IoT equipment production data (operational status and performance) and health data (directly measured).

In this case study, the IoT equipment's digital twin reflected full equipment status (while the digital agent reflected additional data as above, discussed below).

5.4.2 Supervisory Layer

The supervisory layer hosts the digital agent of the system, as well as the system definition, operations data, and control (configuration, monitoring, and control) in the broader sense. These entities have a higher authority level and have access to more status data regarding the system and network. From this layer, it is possible to implement system automation using workflows. The digital agent is focused on the IoT equipment as well as the interconnecting interfaces connected to the IoT equipment.

The digital agent, for testability, may typically be implemented as an "object" in a programming environment with status data (from the field) and associated procedures. The digital agent, in this case, was based on a centrally connected IoT device with field devices connected by wired interfaces and ad hoc wireless networks. The digital agent will utilise on-site resources, field-service agents, or remote technicians from the IoT equipment provider to restore the system to "state 0", which is the fully functional state. Escalations are also handled and linked to maintenance, repair, and support calls with relevant Standard Operating Procedures (SOPs).

5.4.3 Management Layer

The management layer contains the network administration and/or a higher authority manager that monitors other digital agents contained in the total enterprise network (many sites with digital agents). This layer ensures efficient resource planning, resource allocation, and field management according to their individual business rules. The management layer also manages changes to operational and reporting rules to respective digital agents – that is, orchestration at a higher, or enterprise, level.

The human or digital management system monitors digital agents' reports while managing exceptions, such as persistent system/equipment faults or the inability to restore a specific field system after repeated attempts. The management system uses intelligence data from status data to notify engineers or specialised maintenance teams to address the fault reported.

The system proposed above was conceptualised as a method to automate IoT data from multiple sources and was not fully implemented by the Pivot maintenance entity at the time. Elements of such a structure were used to monitor the communication network, Pivot operational and health status, and failures. Deviation from set points, for example, was not explicitly measured and acted upon, which may be addressed using the high-level orchestration in the hierarchy discussed above.

5.5 Data Analysis and Results

The data collection phase of the case study revealed the impact that IoT had on a Pivot system concerning operational, support, and maintenance levels. A detailed comparison was made between the field system both with and without IoT enabled on-site equipment. A summary of the incident types and their individual root-causes is presented as follows:

1. **Electrical Supply Failure:**
   Electricity provider (ESKOM) supply failure, incorrect wiring (phases not connected or incorrect phase sequence), load-shedding, power fluctuations in the power supply, short circuits, lightning strikes. These failures were observed by using the IoT capabilities that include a battery backup, specifically to allow operation of units when grid power fails. These failure types, excluding load-shedding, fluctuations, and supply failures, required a site visit to rectify. In a single case, an electrical contractor was needed for a site visit (swap out) after a lightning strike caused a loss of communication in the system as detected and reported by the monitoring system.

2. **Water Supply Failure:**
   Incorrect water supply system design, pump system failure, valve failure, water line blockages, and losses in the water line causing pressure drops. These metrics were monitored using the IoT enabled power flow and water pressure monitoring system.

Failures of this nature required extensive system design audits, replacement of pump equipment (due to failure) and the removal of obstructions in the water lines. These obstructions in the water lines and leaks causing pressure loss could be rectified by the end-user team without needing a site visit.

3. **Pivot Structure Failure:**

   Tower motor drive failure, safety system failure, wet condition failures (loss of traction on drive wheels), and structural (mechanical) failures were also monitored. The Pivot control system observed these failures by detecting dynamic anomalies from measurements, such as rotation angle, movement speed, and safety return signals. For this class of failures, rectification was possible for the end-user team to rectify without needing a site visit due to the availability of information.

4. **Pivot Controller Failure:**

   Configuration errors, equipment module failure, sensor failure, communications failure, and software malfunctions were monitored and rectified using the Pivot control system. These failures were observed by monitoring the system configuration status, operational status, and system event logs provided by the IoT system. Changes in configuration and software did not require site visits, but any failures in hardware or equipment required a site visit for rectification. There were no cases of critical software failure as all failures were remotely fixed (full stack rectification, including operating system faults), and configuration changes were managed accordingly.

5. **Communications Failure:**

   Failures occurred in both the network layer and device layers of the IoT network. In the network layer, communication failures, slow communication speeds, and power failures. These failures were observed by software that was designed specifically to manage the network layer. Power failures that occurred on-site resulted in communications being lost, while slow network and communication speeds were attributed to network availability and coverage. Antenna changes were made to improve speed as the management system clearly indicated anomalies that could be addressed. Antenna changes required a site visit, but the number of visits was reduced after the installation had been addressed.

6. **Operator Failure:**

   End-user operation error, system tampering, incorrect system use, and incorrect system configuration. These failures were observed using monitoring software implemented in the middleware and application layers to detect changes in system status and consequent system faults. Site visits are not required as rectification can be done by providing end-user support.

7. **System Performance:**

   Irrigation performance, total functional operating hours (uptime) and non-functional operating hours (downtime), and failure frequency. These failures were observed using the status monitoring capabilities of operational performance provided by the IoT component in the system and comparing the status with a set of predefined desired values. Fault rectification did not require site visits, but end-user notifications could be provided.

8. **Configuration and Reconfiguration:**

   After system installation and fault rectifications are completed, the system reconfiguration can take place completely remotely, eliminating the need for a site visit specifically for system reconfiguration. Testing can take place in the manner of the current configuration being verified against previous configurations or a set of predefined values.

It is evident that IoT in the system significantly increased the efficacy and efficiency of corrective maintenance events by reducing the required number of site visits to a single site visit in case of hardware failure and none in case of a software failure or malfunction. This reduction in site visits and increase in efficacy and efficiency of the corrective maintenance events IoT consequently reduced the total downtime of the system and the total time required for fault detection, isolation, preparation and rectification.

The IoT element included in the Pivot system made it possible to provide system performance logs, operational logs, and technical logs that could be used in the preparations for scheduled preventive maintenance events. These logs were made available in the middleware layer to the relevant parties in order to prepare effectively for a site visit. The performance and operational logs could also be used to identify any anomalies or faults that occurred, while the technical records could provide historical maintenance and service data.

5.6 Impact of IoT on Maintenance

The highest impact of IoT on testability was observed during maintenance of the Pivot system due to its ongoing nature (as opposed to the production testing, which is a single event). In the eADR process, the need for a multi-layered digital twin and agent was identified in the system. In addition, an in-depth analysis of the impact of IoT on the system commissioning, corrective maintenance cycle, and preventive maintenance cycle was made. The effect of IoT on testability can be described best by referring to the scope of analysis as shown in captured in figure 4.3.

5.6.1 Downtime

As with any system, the goal is to minimise total overall downtime. The addition of the IoT component indirectly allows downtime causing actions/processes to be performed during uptime. The update of control software is one such action. Without an IoT component, a site visit must take place, the system must be shut off, and a new software version must be uploaded. With IoT, the update of system software can occur remotely during system standby, removing the need for system shutdown, with a full digital twin allowing the system to keep its technical configuration and operational settings without human intervention.

Measuring actual downtime in the study proved impossible, and inferences were made from observations and historical visits. The variance between the different Pivot system installations was simply too high. The environmental factors each individual Pivot unit faces, along with customer involvement and habits and operational conditions, made it very difficult to determine accurately measured downtime of a single unit and will need to be done on a case-by-case basis in the future - this must follow from information obtained in the orchestration layers discussed above.

What could be identified was the biggest cause of downtime and the impact IoT had on the case study Pivot systems in terms of handling and addressing downtime. At the time of this study, electrical failures of the Pivot had the highest contribution to system downtime, although electrical failures only affected around 1% of all Pivots. Electrical failures mainly caused by large spikes in the electrical supply grid, lightning, and human error caused the Pivot to fail and halt operation. At the time of electrical failure, the IoT component of the Pivot remained operational and was immediately reported to the cloud for replacement.

Without the addition of IoT the reporting of the fault would not be possible and the total downtime of the system would have increased significantly (a more detailed and quantified analysis follows below). The intrinsic early detection capabilities that IoT provided reduced the downtime as the detection and isolation of the fault had already been completed almost at the time of failure. In the case that the system required the replacement of electronic parts or devices, IoT connectivity allowed remote reconfiguration of the new devices and restoration of the system settings to those before the fault had occurred. Restoration of previous configurations was possible by employing all historical data stored on the cloud.

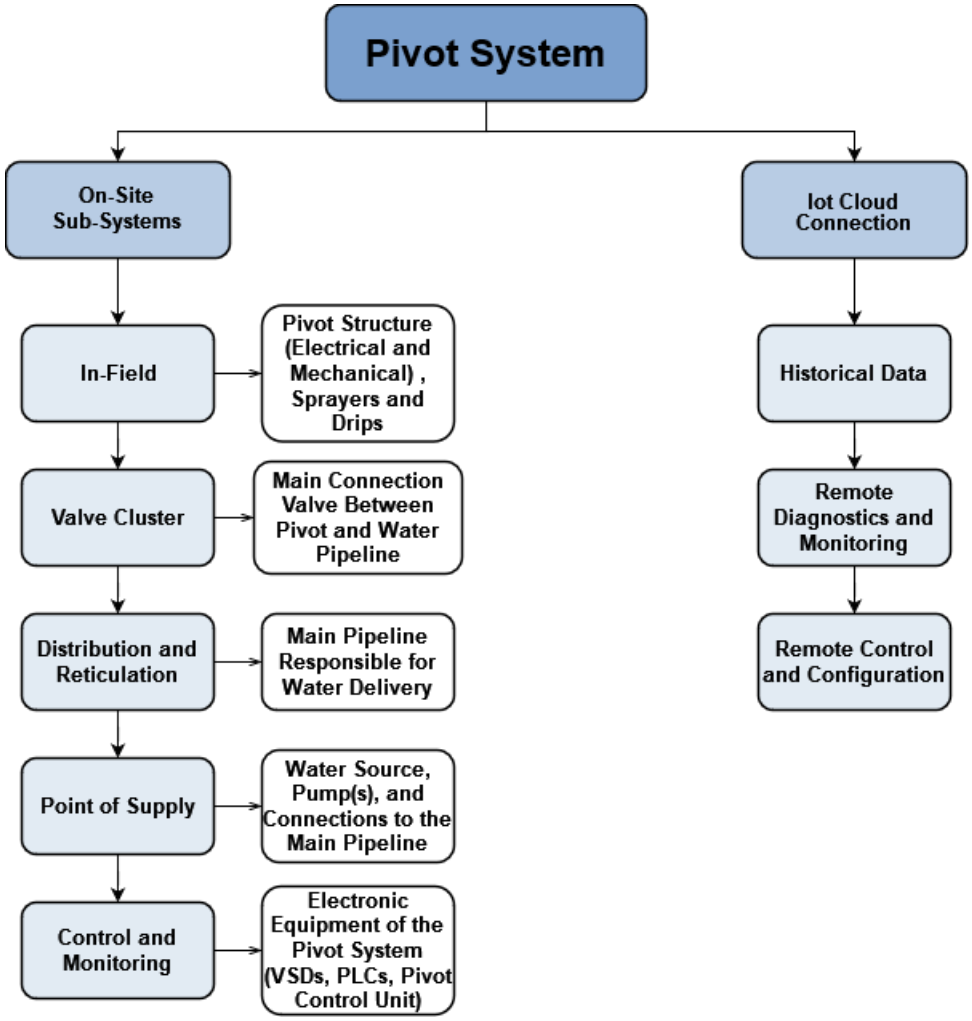

Figure 5.7: Sub-Section Breakdown of the Pivot System

Figure 5.7 captures the Pivot system breakdown. To analyse the downtime in more detail, the Pivot system was broken down into five sub-systems that could individually be assessed in terms of the administrative delay -, logistics delay -, and active maintenance time.

The IoT cloud connection was individualised in this breakdown to identify the different functions it provided to the system. These functions can impact the system in many different aspects and cannot be explicitly discussed in terms of administrative- , logistical- , or active maintenance delay time. The impact of the IoT component is discussed as it affected a specific delay time, as discussed below.

5.6.2 Administrative Delay Time

The administrative delay time contains all administration activities that must occur during any action or process performed either for or on the Pivot system. Actions such as fault detection and fault isolation were classified as administrative delays and will be the main focus points. According to these actions, the delays of the five sub-systems of the Pivot could be estimated based on historical service data.

- **In-Field:** The elements in the in-field category are either directly connected to the electronic control of the Pivot system or their status can be inferred from the captured data. When focusing on the in-field elements, fault detection can occur almost immediately.

  On the other hand, the isolation of the fault could take up to three days. The delay comes as a result of the limitation in the monitoring of the system. Mechanical elements, such as motors, are currently monitored indirectly. In other words, their status can be somewhat inferred from monitoring other elements. This indirect monitoring, in some cases, results in longer administrative delay times.

- **Valve Cluster:** Contained in the valve cluster is the main valve that connects the main water pipeline with the Pivot. The only point of failure is an electronically controlled solenoid valve. The detection and isolation of a fault in the valve cluster happens quickly and does not cause any quantifiable administrative delay.

- **Distribution and Reticulation:** This sub-system can fail in one of two different ways, although both failures can occur at the same time. The failure can be either a leak or a burst in the pipeline. In the case of a burst pipeline, the fault detection will be immediate as the electronic control will notice the water pressure is absent and will be able to alert as soon as the failure occurs.

The isolation of the fault can take up to 12 hours in some cases. In the case of a leak, the detection of the fault will be lengthy and can take up to one week to detect. The detection is only possible in such a relatively short time frame because of the IoT component. Historical data regarding the water pressure can be used to detect a gradual drop in the water pressure, indicating a leak. Functionality that is not available in a system that is not IoT enabled.

- **Point of Supply:** The main focus element in the point of supply is the water pumps. Different components in the water pumps are prone to wear and need to be replaced over time. Detecting a fault in these components can be difficult as the failure of these components is most frequently not immediate and happens gradually. To detect a failure in the pump components, the pressure needs to be monitored. The implementation of IoT allows the analysis of historical data to identify gradual drops in pressure. Isolation of the fault to the pump components can take between two and six hours.

- **Control and Monitoring:** The control of the system is almost negligible in terms of administrative delay. The Pivot control unit is connected to the cloud. It can report any faults immediately after its occurrence and provide detailed diagnostic data identifying the cause and area of the fault. For example, in the case of Variable Speed Drive (VSD) motor speed control devices' failure, the pressure will fall and be detected as a failure. The isolation of the fault is relatively quick as it requires a visual inspection to identify the VSD as the failure point.

5.6.3 Logistic Delay Time

The logistic delay time contains all logistic activities that must occur during any action or process performed either for or on the Pivot system. These actions included part acquisition and site visits which will mainly be focused on. Logistic delays can be estimated according to these actions for each of the sub-systems.

- **In-Field:** For the in-field elements, very few logistic delays exist with the addition of the IoT functionality. In the case of part acquisition, all parts are readily available, quick to restock, and do not add significant logistical delays. In terms of site visits, around 70% of failures can be addressed and restored remotely via phone call. The addition of the remotely available data provided by IoT further reduces the total time required to address the failure to a maximum of two hours.

- **Valve Cluster:** The valve cluster is a simple and robust sub-system in the sense that it consists of very few elements that can cause failure. The parts prone to failure are readily available and easy to find replacements locally. Repairs performed via phone call have to date had a success rate of 100% and completely eliminated logistical delays.

- **Distribution and Reticulation:** The main pipeline is a purely mechanical structure without any form of electronic control or monitoring. The two ways of failure, burst or leak, for a small section of the pipeline call for similar repair times and part lead times of around one day. If the entire pipeline must be replaced due to failure or damage, a combined lead and repair time of two to three weeks can be estimated. These lead times are estimated from the day of order to the day of delivery on site. Remote repair for this sub-system is impossible, although site visits are rarely needed.

- **Point of Supply:** The pump(s) included in the point of supply are the main focus in terms of logistical delay. The failure of pump components, such as mechanical seals or impellers, happen gradually and can easily be diagnosed. Remote repair by phone is possible around 50% of the time, otherwise requiring a site-visit. Lead times for faulty components are estimated at around four days with site-visit repairs taking up to two days.

- **Control and Monitoring:** In the case of the electronic control and monitoring systems, site visits are required around 80% of the time, causing a significant addition to the overall logistic delay. This is because remote repairs by phone prove difficult to users with little to no experience in working with these electronic devices. Remote rectification can be attempted with IoT enabled devices but is not possible otherwise, and a site visit would be required. Replacement of these parts is within three to five days in the case of a critical failure.

5.6.4 Active Maintenance Time

Active maintenance time can be divided into system commissioning, corrective maintenance and preventive maintenance. Analysis of the Pivot system can be done in accordance with these categories to precisely identify how the IoT component impacts each of the maintenance cycles and system commissioning. Possible areas of improvement can be identified in these categories, and recommendations can be made on changes, additions or adjustments.

1. **System Commissioning:**
   Initial commissioning of the Pivot systems only occurs at the beginning of the system lifetime or when major upgrades occur. During the commissioning of the systems, IoT data can be used to aid the process. The data captured from previously commissioned systems assists in preparation for the commissioning actions. Environmental data captured from different Pivot systems in the local area can be used to determine what the commissioning environment would look like before arrival. In the case that specific equipment would be needed for certain environmental circumstances, e.g., recent heavy rain, the relevant preparations can be made before the commissioning site visit.

   During commissioning, the Pivot must be tested, calibrated, and verified. Initial system testing entails measuring the power supply, water supply pressure, mechanical functionality (in gearboxes, motors and axles), and connectivity (local and wireless). With the enabled connectivity and IoT component, the system can be fully tested remotely with very limited (almost no) human intervention. Calibration and verification of the system can take place remotely by using the measurements captured with IoT and observational feedback from the field. The measurements can be compared to a predefined set of values and adjustments can be set in the system configuration.

   System verification can be done by observing the calibrated system and comparing the data to other Pivot systems in the local area or similar areas. The observations can then be used to ensure the newly commissioned system is functioning as expected. These capabilities provided by the IoT component allowed the fault-finding process during the commissioning stage to have a minimal impact in terms of time. The reason is that the fault can be isolated and localised by using the IoT functionality, simplifying the fault correction process and significantly reducing the labour time spent on fault finding and troubleshooting.

2. **Corrective Maintenance:**

The current Pivot system is mainly supported by reactive maintenance. The information system (please refer to the discussion on high-level orchestration layers in the system hierarchy) currently does not implement RCM as limited CBM data is available. This will be highlighted when analysing the system's preventive maintenance and potential improvement. This means that the current maintenance philosophy of the system is RTF with preventive maintenance being done by the end-user on specific components. Better end-users will thus fare better in this area. Components or equipment, when supplied by the client to the end user, can thus be allowed to operate up to the point of failure before replacement occurs. Alternatively, they will be replaced when the running hours of the system exceed predetermined thresholds - this was left to the end-user to manage. This approach to maintenance can be improved through automation as costs associated with RTF-type maintenance is typically higher than predictive approaches.

The Failure Rate (FR) of the system was estimated for five identified sub-systems as shown in 5.2. Estimates for the FR of the sub-elements of the system were based on the reported failures from end-users and identified failures by SMEs. Their root causes were recorded by the IoT company over the lifetime of the Pivot systems. These estimations identified possible areas where maintenance could be reviewed to improve the system performance. When considering the whole system, the in-field sub-system FR was estimated at 35%, the valve cluster at 5%, the distribution and reticulation at 5%, the point of water supply at 25%, and the control and monitoring at 30%. It is important to note that these were estimates based on observations and measurements only for the subset of end-users referenced in this study. These findings can not be generalised but were found to be applicable and instructive within this research scope.

Individual IoT Pivot systems currently report operational data, high-level diagnostic data, and current system status. This information is then interpreted by an operator/supervisor which arranges a maintenance-focused site visit to correct the fault if it is detected. At this stage, the IoT element is well implemented as it can detect and identify, in most cases, the fault that occurred. This simplifies the preparation process by providing information regarding the faulty part, environment, possible fault cause (in some cases), the operation executed when the fault occurred, and time the fault was detected.

IoT is present in the corrective maintenance cycle of the Pivot system, but the application of this data may also enable predictive maintenance. Automated maintenance functions can possibly estimate future failure time from a component's reliability data. This could reduce the number of unexpected failures and downtime as replacement parts may be obtained and replaced before a system failure occurs. This may increase the overall efficacy of maintenance.

3. **Preventive Maintenance:**

   Currently, no active preventive maintenance is implemented by the client in terms of SM, CBM, or PdM since the end-user owns that responsibility. That is, the end-user uses system data to perform active preventive maintenance as part of an annual maintenance process. Depending on end-user notification, major adjustments and other actions are currently applied in the Pivot systems. The end-user notifies the client of changes required because of changes in desired operational behaviour or significant environmental changes. These changes would then require the Pivot system to be re-calibrated, reconfigured, or upgraded with new equipment.

   The IoT Pivot system is relatively new to the market and has not been able to gather sufficient data in the timeframe to implement PdM practices effectively. Currently the Pivot system has aspects of preventive maintenance actions implemented, but a specific focus on preventive maintenance as part of the system's high-level orchestration will be beneficial. An example may be monitoring water flow in the water distribution lines. A gradual decrease in flow (and pressure) could be an indication of either slowly increasing leakage in the distribution line or pump seal wear. Such a fault can be identified early, before failure of the parts, and maintenance can be performed prior to failure in preventive maintenance (historical data analysis).

   Preventive maintenance actions that the Pivot can implement at the moment without significant change to the system infrastructure is to apply available data effectively, and to add more IoT sensors potentially. Considering the FR of the point of supply, closer monitoring of the pump(s) may result in less downtime caused by failure. Possible metrics may include flow rate monitoring of the water entering and exiting the pumps to identify possible water source obstructions or pump malfunctions, while pump temperature and vibration sensing can be used to detect wear on pump components.

Measurement of pressure ripple in the distribution pipeline may be used to identify cases where the distribution line is more susceptible to bursts or leaks. The identification of exceptions can thus enable timely provisioning of parts.

PdM will lead to the application of more complex parameters such as optimal maintenance time, crossover time, and RUL of components and sub-systems. When these parameters are used alongside a digital agent, an almost fully automated system can be created. The Pivot system calculates these parameters and notifies the relevant parties of when it predicts these points in time will be reached. In the case of optimal maintenance time, the Pivot can automatically schedule maintenance action and document which parts require attention.

In the case of the crossover time, relevant parties can be notified. At this point, the responsible maintenance party can change the maintenance approach for the specific Pivot from preventive to corrective, which is how most end-users in this sector operate in South Africa. The RUL can also be included in the predictions, provided sufficient CBM data is available, that can be provided to the responsible maintenance party to make adequate preparations before the RUL prediction indicates part failure.

5.7 The Impact of IoT on Testability

At this point, reflecting on the systematic literature review and results from the case study and the life cycle analysis is necessary.

5.7.1 Reflection on Literature and Findings

The table below presents a reflection on how IoT affects testability. Different actions in the Pivot system life cycle were identified to show how IoT impacted the Pivot system's effectiveness. Table 5.1 shows the comparison between these actions with and without the impact of IoT.

Table 5.1: Comparison of Pivot System With and Without IoT

Pivot Without IoT	Pivot Characteristic	Pivot With IoT
Requires multiple scheduled site visits. Manually monitoring such a system long-term also requires hardware capable of storing data until the following data retrieval.	**Long-Term Monitoring**	Requires no site visits unless the system is offline. Data can be retrieved remotely at any point in time at virtually no cost.
Impossible to detect the possible fault immediately. Fault detection can occur only after the system has halted operation. A site visit is required in some cases.	**Fault Detection**	Requires no site visits unless fault cannot be detected, reported, or the system is offline. The system can automatically report any faults and notify the relevant parties. This also results in minimal system downtime.
Requires a minimum of one site visit. Multiple site visits can be required if the identified fault requires specialised equipment to be restored.	**Fault Identification and Restoration**	Site visits may be deemed unnecessary unless the system is offline. The fault can be identified remotely and either be restored remotely, or a single, well-prepared site visit can be made. Time spent at the site will also be reduced as the fault has been identified at the time of arrival.

Continued on Next Page

Table 5.1 – Continued from Previous Page

Pivot Without IoT	Pivot Characteristic	Pivot With IoT
Requires in-person data capturing and recording. The data available can become outdated fast. Expensive in terms of time and labour.	**Diagnostic Data**	Data can be updated as frequently as desired while being readily accessible to the user at any time if the system is not offline. Virtually no costs to access the data.
Maintenance record is to be captured in person. Can become inaccurate over time or completely lost. Inconvenient to the user and maintenance personnel.	**Maintenance Data**	Detailed data of the maintenance history can be kept for as long as desired. Allows the incorporation of maintenance notifications when maintenance is needed and predictions when maintenance will again be needed.
Requires site visit at every reconfiguration. Reconfigurations can occur at any point in time, depending on the site. Changes in environmental conditions, for example, can require a system reconfiguration.	**System Configuration**	Requires no physical site visits unless an in-depth reconfiguration is required or the system is offline. The system can be reconfigured in any way remotely at any time at virtually no cost

Continued on Next Page

Table 5.1 – Continued from Previous Page

Pivot Without IoT	Pivot Characteristic	Pivot With IoT
Monitoring and control are only possible when on site, which requires travel and time. The system status is not available remotely, leading to higher operating costs in terms of time and labour.	**User Operation**	Requires no site visit or physical human interaction with the equipment unless the system is offline. The user can fully access, control, and monitor the system remotely.

5.7.2 Resource Management

IoT impacted every aspect of the system in terms of downtime. The impact becomes visible when comparing different function-resource relationships when IoT is both implemented and not. To create such a comparison, the most critical actions performed during system downtime, called functions for the sake of comparison, were identified.

Similarly, the most important resources required by these functions were identified and the relationships between them were captured in table format. This resource allocation table allows for visual comparison of IoT impact on the Pivot system.

Individual metrics allow the different aspects of system downtime to be combined to produce a single metric that can be used to compare the Pivot system with and without IoT. The IoT Pivot system benefited from the centralisation of maintenance-critical resources. This was confirmed in the process of analysing the function-resource relationships. The centralisation of costly specialist resources significantly impacting factors such as the overall logistical delay and active maintenance time. Consequently, the impact on these factors had an overall impact on the total downtime the system experienced, which table 5.2 demonstrates. Figure 5.8 shows the contrast between a system environment with centralised and decentralised resources.

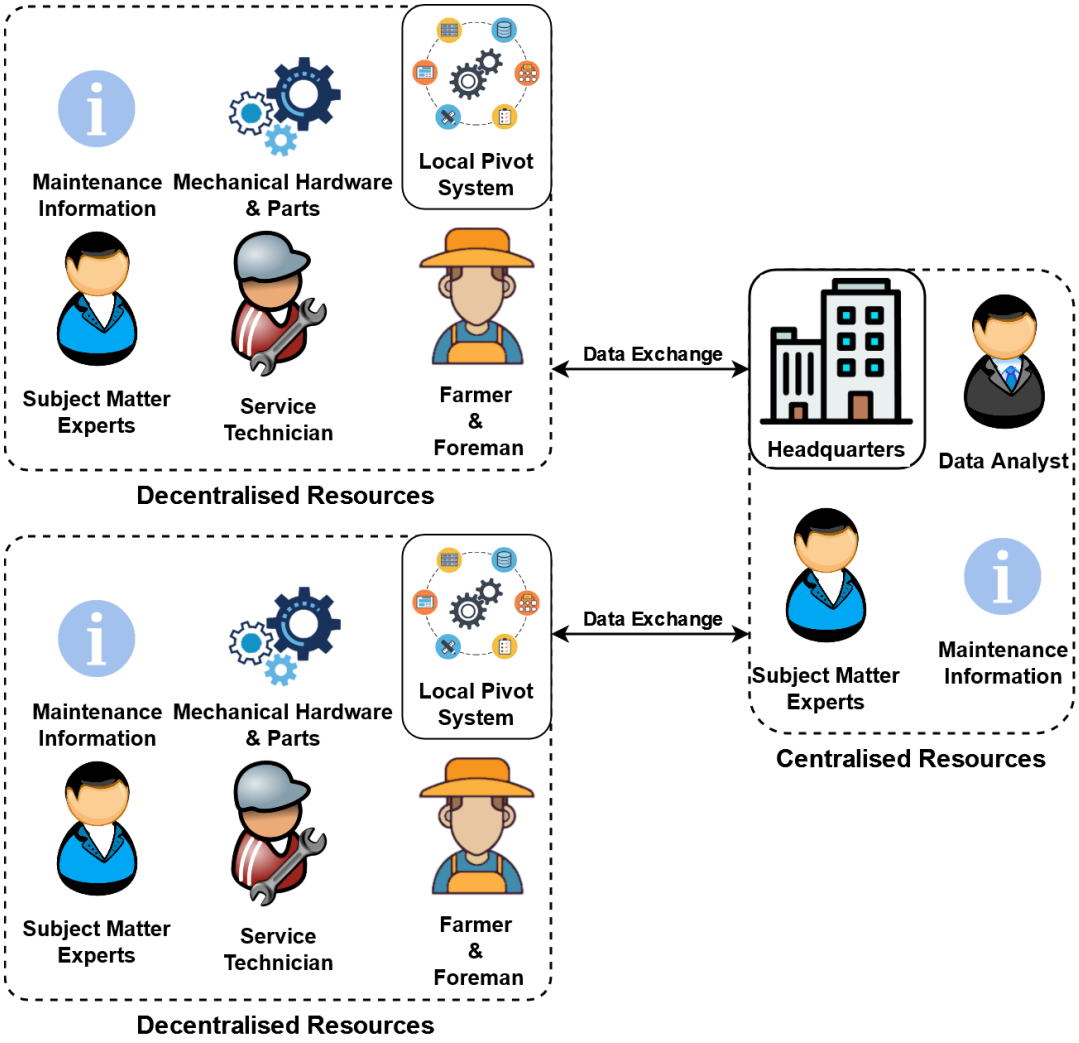

Figure 5.8: System Environments with Centralised and Decentralised Resources

The decentralising nature of IoT results in a hybrid resource model with critical and non-critical elements appropriately allocated. Maintenance-critical resources were centralised, while critical operational resources remained decentralised (localised to farms). Resources that are both on-site and centralised will be referred to as hybrid resources, for example, mobile subject matter experts. A system resource hierarchy can be created according to these resource classifications as captured in figure 5.9.

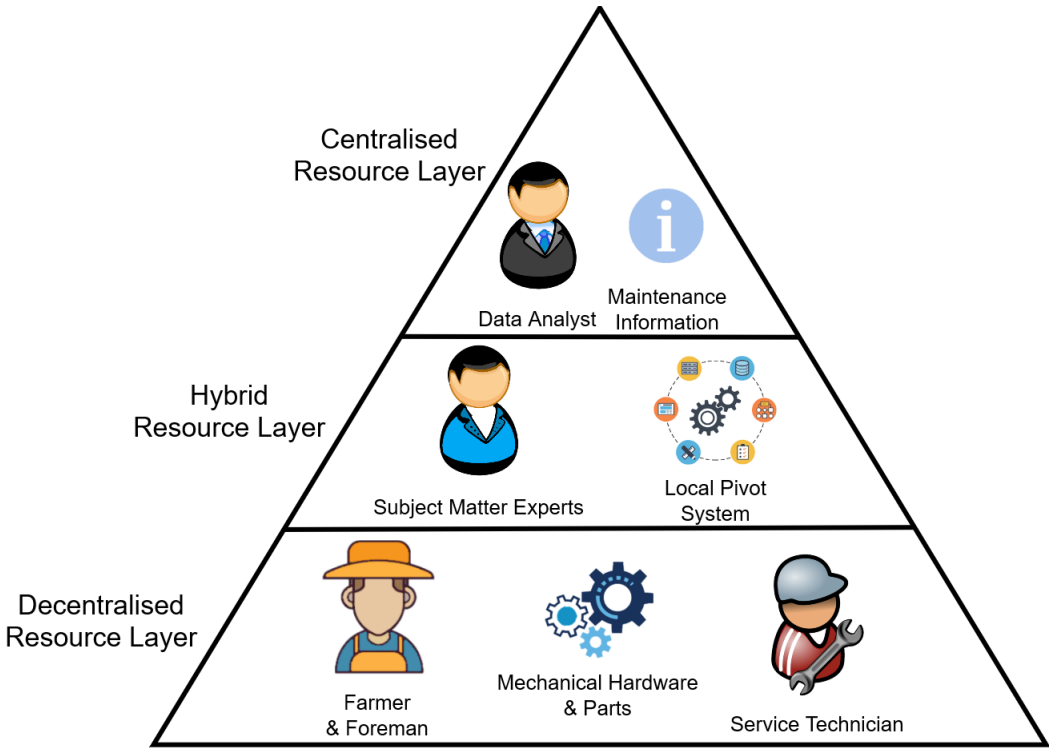

Figure 5.9: Hierarchical View of Resources in the Hybrid Resource Model

The figure shows that the resources local to each Pivot system (decentralised resources) are captured at the bottom of the hierarchy. The hybrid layer employs these bottom layer resources to collect data regarding the Pivot system status. The Pivot system and the subject matter experts are captured in the hybrid layer as these resources are active in both the centralised and decentralised environments and make the connection between the centralised and decentralised resource layers. The final (top) layer containing the centralised resources utilises the information captured by the hybrid layer to conclude the Pivot system maintenance activities.

In the resource allocation table, the scope was limited to the IoT factors associated with downtime. It is assumed that the Pivot system will always have access to a reliable network connection, which was verified in practice as a highly reliable managed network provided by the IoT company. Resources in the table included specialised equipment and subject-matter experts and operators. Specialised equipment was defined as expensive and scarce equipment that are not typically included in an average technician tool set. Specialist interaction was defined as the action of a human directly interacting with the Pivot system's hardware without a remote device. With these definitions and assumptions, the function-resource comparison was created and captured in table 5.2.

Table 5.2: Function-Resource Comparison With and Without IoT

		Resources											
		Without IoT						With IoT					
		Logistic Actions	Labour Hours	Subject Experts	Specialised Equipment	In-Person Interaction	Centralised Resources	Logistic Actions	Labour Hours	Subject Experts	Specialised Equipment	In-Person Interaction	Centralised Resources
Functions	Maintenance Preparation		🔴	🟠					🟠	🟢			🔴
	Fault Rectification	🔴	🔴	🟠	🟠	🔴	🟢	🟠	🟠	🟠	🟢	🟠	🟠
	Fault Detection & Isolation	🔴	🔴	🔴	🟠	🔴		🟠	🟠	🟠	🟢	🟠	🔴
	System Validation & Verification	🔴	🔴	🟠	🟢	🔴	🟢	🟢	🟢			🟢	🟠
	System Configuration	🔴	🔴	🟠		🟠	🟠	🟢	🟢	🟠		🟢	🟠
	System Monitoring	🔴	🔴	🟢	🟢	🟠		🟠	🟢	🟢	🟠		🔴
	System Logs	🟠	🟠			🔴			🟢				🔴
	System Diagnostics	🔴	🔴	🟠	🟠	🔴			🟢			🟢	🟠

🔴 Unavoidable

🟠 Conditional

🟢 Rarely

It is evident from table 5.2 that IoT impacts all critical aspects of maintenance functions. When comparing the function-resource relationships from both sides of the table, it is clear that IoT reduced the overall resource requirement of the different maintenance functions. A significant difference to note is the reduction in the resource dependency of the different functions. This is due to the centralisation of resources, such as the SMEs, and the availability of historical and real-time system data.

An increase in system dependency on centralised resources was identified from the table. The centralisation of resources shifted the system dependency from multiple resources to a single resource, making the centralised resources a high-impact point of failure that must receive attention in design and business continuity.

The table shows a clear contrast between the system with and without the IoT component. A brief discussion regarding the system resources in accordance with the table is as follows:

- **Logistic Actions:** Without IoT the overall system dependency on reliable logistic actions is higher than with IoT. Less functions can be performed without logistic effort due to lack of information, the ability to interact with systems on-site, and lack of coordination remotely. This is not the case with IoT, and the improvement lowers a significant cost component, namely logistic planning, coordination, travel, and opportunity cost.

  A significant improvement is that system maintenance data and subject matter experts are shared in the IoT centralised case. Often, IoT is only considered to improve access to data, and the focus is not on operational and maintenance impacts - this analysis clearly highlights the improvements associated with reliable IoT.

- **Labour Hours:** A lack of IoT functionality causes a high number of unavoidable labour hours in almost all maintenance related activities. The decentralisation of resources caused by the absence of an IoT component leads to elementary and redundant tasks being completed by humans that could otherwise apply their time in more productive ways. The number of labour hours spent on maintenance functions is significantly reduced with IoT. From the analysis, it is evident that the effort to collect data is notably reduced, remote configuration of sites can be done, and expert input is more effectively applied.

- **Subject Matter Experts:** The dependency on SMEs relates to the dependence on logistic actions and spent labour hours. Increased reliance on SMEs would, without IoT, also increase logistic effort and labour hours.

IoT enables subject matter experts to act as hybrid resources. Experts, in some cases, will still be required to act as a decentralised resource but will more frequently act as a centralised resource. IoT shortens the time required for the collection, analyses, and action tasks and adds the ability to process large amounts of data for decision support. From the analysis, it was evident that reliable maintenance data often eliminated subject matter expert effort due to the big data processing ability associated with IoT. For example, tedious fault-finding tasks were often eliminated based on historical site and equipment performance data.

- **Specialised Equipment:** The need for specialised equipment in a system without IoT is slightly increased with respect to IoT systems. Specialised equipment is required in specific cases where access to humanly invisible information is impossible - for example, to see power line and water pressure status. The absence or logistic delays of specialised equipment will significantly increase downtime from a testability perspective.

  With the incorporation of IoT, the Pivot system itself becomes the specialised equipment for in-field applications. The Pivot system implements all critical measurement functions and with IoT implements the test philosophy's principle of testability in the true sense of the test philosophy, namely to measure, compare, and report. A reduced dependency remains for specialised equipment used for measuring mechanical sub-systems.

- **In-Person Interaction:** In-person interaction will be used to obtain on-site information through site visits, phone calls, and other forms of information acquisition effort. The dependency on in-person interaction with the Pivot system is high when implemented without IoT as system maintenance data remains decentralised, which also adds to logistic delays and labour hours.

  Having access to maintenance data with IoT reduces in-person interaction with the Pivot operators and owner, as most of the interaction with the system has been automated. Data collection and control tasks are automated, and error-prone interaction with on-site personnel and equipment, which is often tedious and stressful to participants, is eliminated in most cases.

- **Centralised Resources:** The centralisation of system resources in a system without IoT connectivity is limited to manual functions, and access to both real-time and historical information is restricted. The system is still dependent on limited centralised manual resources as most resources will be in the field. Still, the absence of centralised resources would not affect the system in a significant way. Interestingly, centralised human resources training, data processing, and coordination are required and introduce a business continuity risk.

  From the analysis of the maintenance workflow, IoT in the system increases the dependency on centralised resources significantly. With IoT in the system, a single, high-impact point of failure is introduced and all efforts must be made to ensure the availability of this critical resource. The creator of the Pivot system, the client, assured availability by design, and the "industrial" nature of this system thus emphasised the need for good Information and Communication Technology (ICT) design and availability of cloud and network systems. Importantly, this dependency also resulted in operational functions being localised, which moved operational decisions to the in-field systems for autonomous operation.

What is critical to note, and of very high impact, is that the centralised nature of the IoT system considered in this case study, creates a single point of failure. This vital consideration was addressed using an approach where: (i) all critical on-site equipment remained autonomous in nature and no operational decision-making or execution control took place in the cloud, (ii) dual redundancy and failure mitigation were put in place to ensure access to site information was managed under challenging conditions such as load shedding, and (iii) important information on all sites was accessible to subject experts and other decision-makers for the purpose of maintenance while all critical operational decisions could be made without access to the cloud. Hence, the table represents a system where the dependency on the centralised functionality of IoT has been taken into account.

From the table, the reduction in resource use by the system is qualitatively significant and is indicative for the Pivot system as it currently stands. Completing the function-resource relationships and comparing the system with and without the IoT component also revealed functions where IoT assistance could be improved. Fault detection and isolation, fault rectification, and system monitoring remain the functions that require the most resources. This can be attributed to the current lack of monitoring mechanical systems and the extent of IoT-enabled devices.

5.8 Cost and Time Impact of IoT

The real-world benefit that IoT provides to a physical system is analysed based on observations and high-level measurements of time allocation and logistic effort in the case study. A notable benefit is a significant improvement in cost-reduction that IoT provides.

Installations on site required visits to different sites, and often these site visits would occur multiple times to complete installations or configuration of devices. The time frame in which the case study was conducted was limited to one year. The values for the time and cost savings can only be illustrated for single visits in this analysis.

The distances for these site visits range from 50 *km* to 400 *km* and thus highlight the importance of savings considerations for remote sites. A fairly accurate estimation of the time and costs required for a single site visit was based on actual site visits and additional maintenance data (from participants in the project).

The following values were used for time and cost estimation for a site relatively close to the depot, with a distance set at 50 *km* for simplicity.

- **Fuel Price:** R 21.61 (at the time of the study)
- **Distance to Site:** 50 *km*
- **Average Fuel Consumption of Vehicle:** 10-15 *km/l*
- **Labour Fees per Technician:** R 105 *per hour* (two technicians per site visit)
- **Average Duration of Site Visit:** 6 *hours*
- **Average Travel Time:** 1 *hours*
- **Cost of Single Site Visit:** R 1,509.06 - R 1,581.10

For a site located 400 *km* the following values apply:

- **Distance to Site:** 400 *km*
- **Average Duration of Site Visit:** 6 *hours*
- **Average Travel Time:** 10 *hours*
- **Overnight Cost:** R 850 (total for both technicians)
- **Cost of Single Site Visit:** R 4,312.52 - R 4,888.80

To quantify the impact of the IoT component of the system, the financial costs per year for the system of a single site were estimated for the site at 50 *km*, and 400 *km* using the previously mentioned metrics and the following estimates from site visits in 2021:

- **Site Visits per year without IoT:** 6 (including fault isolation and delays)
- **Site Visit per year with IoT:** 2 (where planning is done in advance)
- **Mechanical Lifetime of Equipment:** 20 years

With the above values, an estimation of the system's maintenance costs over a 20-year period can be made for each site. These estimations are captured in table 5.3:

Table 5.3: Time and Cost Saving Estimations With and Without IoT

Estimation Parameter	50 *km*	400 *km*
Cost with IoT	R 60,362 - R 63,244	R 172,500 - R 192,552
Cost without IoT	R 181,087 - R 189,732	R 517,502 - R 586,656
Time Spent on Repairs with IoT	12 person-days	27 person-days
Time Spent on Repairs without IoT	35 person-days	80 person-days

In the estimations above, repeat visits, system downtime, wastage, spillage and other losses were not considered specifically. These costs will also increase as inflation and travel costs (to the service provider) increase over time. The cost reduction when using IoT was found to be a factor of around 3. The values captured in table 5.3 are estimations as these illustrate the point.

From these estimations, it is clear where IoT testability affects the full system life cycle, namely the operation and maintenance process tasks. When considering these life cycle tasks, it is evident that the corrective and preventive maintenance cycles are where most testing is done.

5.9 Ensuring Testability

Testability in the system requires measuring the status and performance of the total system in the different layers of the IoT system architecture, comparing measurements against set or historical values, and escalating deviation from set points for action.

A digital agent, in the true sense of IoT, will automate the acquisition of site status and performance data, definition and adjustment of comparative set points from historical and shared system data, and escalations of deviations from expected values to predefined recipients [19]. Such an agent may also implement workflow automation to inform, control, and track rectification progress and thus improve uptime.

Considering the hybrid resource model of the IoT enabled Pivot system, a digital twin and agent will be incorporated in the hierarchy's decentralised and hybrid layers. The digital twin would be captured in the decentralised layer to assist in local data collection and act as a digital assistant to the farmer/foreman. The twin would report the data collected to the digital agent in the hybrid layer.

A digital agent would be located in the hybrid layer alongside the subject matter experts as it is not purely a "digital" function. The agent, acting as a high authority entity, would assist the subject matter expert in collecting and processing data, most likely in the form of an artificially intelligent "expert system". Figure 5.10 shows the implementation of the twin and the agent into the hybrid resource model.

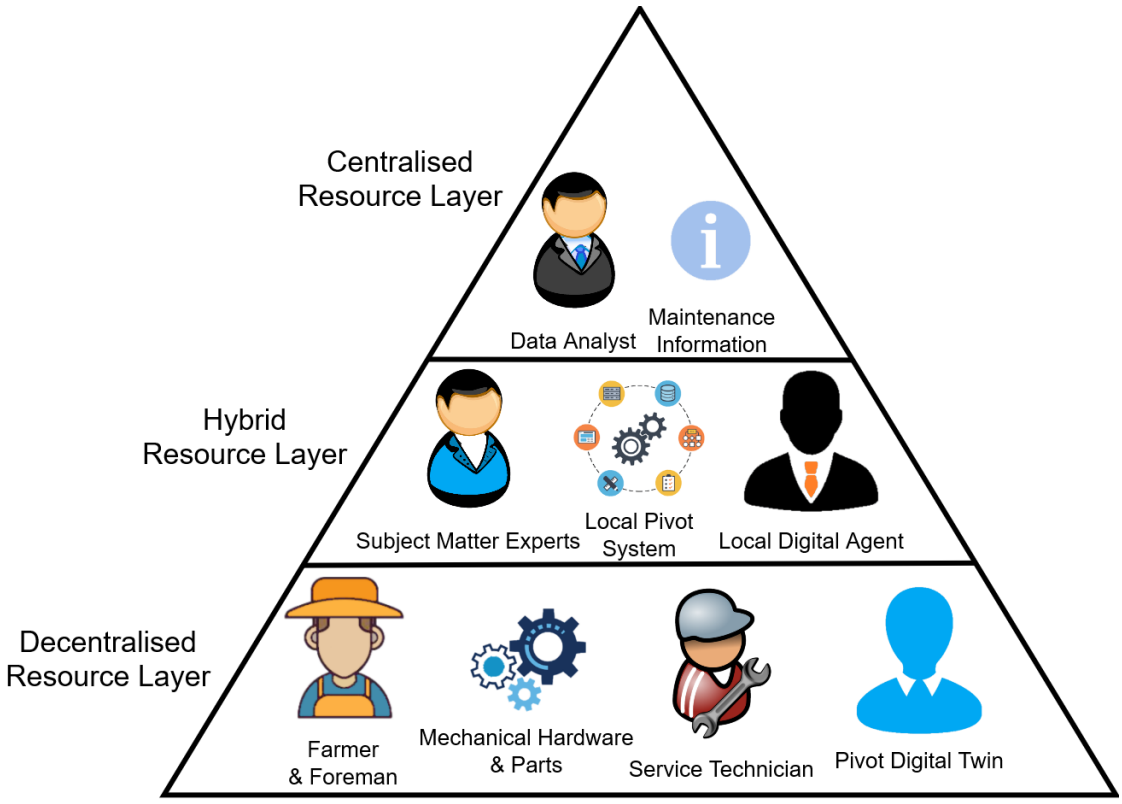

Figure 5.10: Digital Agent and Twin in the Hybrid Resource Model

Thus, the main focus of the digital agent is to monitor the status of different system elements and its operational performance when compared to a predefined set of desired values. The agent is then responsible for providing actionable information as an output. Figure 5.11 shows the layer where the digital agent is incorporated into the IoT architecture.

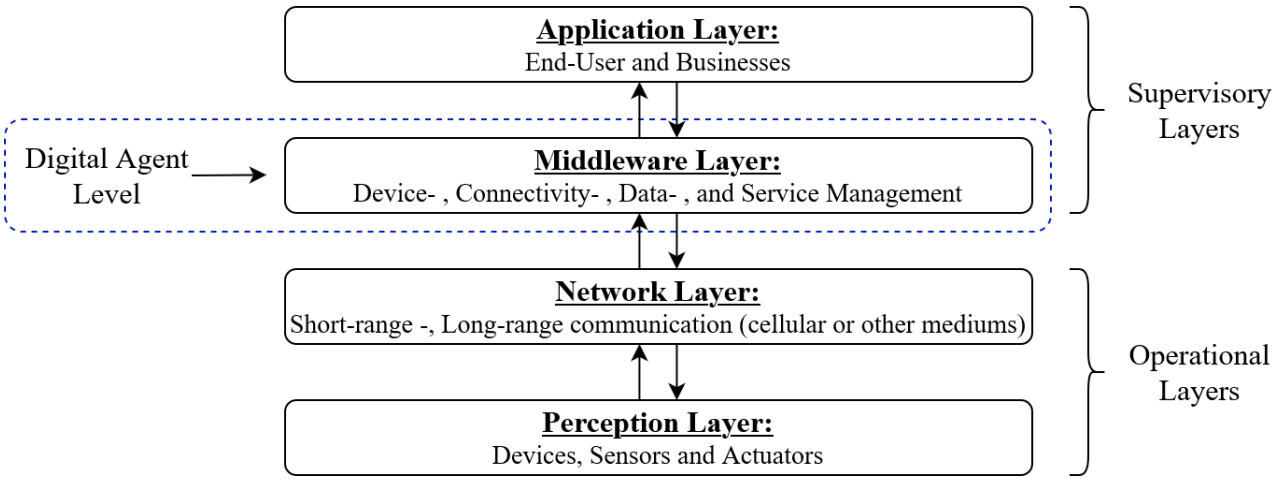

Figure 5.11: Digital Agent in the Middleware Layer

The digital agent in this layer will be structured to enable the system to follow a form, fit and function approach to produce actionable information. The agent implemented in the middleware layer cannot directly access the devices or sensors in the perception layer and is heavily reliant on the functionality of the network layer for updates to the system's status and operational and performance data.

The agent can request predefined set values from a stored configuration and compare the values with the current condition or status of the system and determine the system's current "form". The "fit" of the system reduces to the interconnected network and the system's network configuration. The agent can use this network and the interconnections between the nodes and the overall system to obtain information regarding the environment, system configurations and operational characteristics.

The "fit" also describes how an agent is defined in the middleware layer in the IoT architecture. The agent can by utilising the data captured as the "form" of the system, compare predefined set values to the system's status, and determine if the "function" of the system is as expected. The agent can generate the appropriate actionable data outputs based on the comparison results. Figure 5.12 shows the structure of the digital agent implemented in the middleware layer.

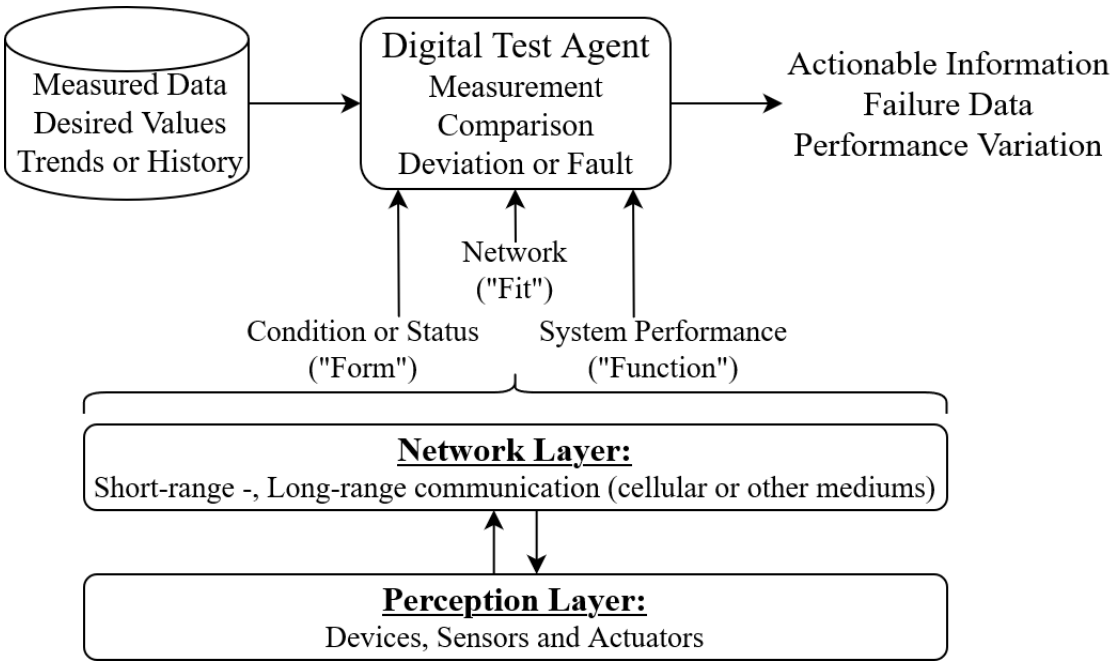

Figure 5.12: Structure of the Digital Agent in the Middleware Layer

A digital agent was defined in the higher-level orchestration layer as an autonomous entity that ensures testability. The case study underlined the importance of an agent due to the highly connected nature of IoT that supports testability in the Pivot system. The characterisation of a physical Pivot site includes the system metadata, historical data, desired reference values, and actionable information used to orchestrate maintenance.

The definition of a digital agent was guided by the eADR process as shown in figure 2.6. The agent in this study is produced as the artefact that eADR follows from the problem analysis phase. This research was conducted in the "problem" entry point in the eADR process. The development of such an agent fell outside the scope of this research, but its definition will lead to a software implementation in future studies.

5.10 Conclusion

The case study identified the need for a high-level orchestration layer responsible for the collection, processing, and presentation of data in an IoT system. Investigation into the implementation of such a layer revealed that this orchestration layer is responsible for the reliable connection between the physical world and the cloud.

Analysis of the data collected from the case study revealed a significant impact on the maintenance cycles of the Pivot system. IoT notably increased the efficiency of corrective maintenance by providing real-time system data. Real-time data reduced the system downtime reducing the administrative-, logistical-, and active maintenance time.

From the case study, essential aspects of the testability of a Pivot system became evident (not evident from the literature study). For example, the importance of IoT in a centralised versus localised configuration came to light. Evidence from the field indicates that IoT network reliability (GSM in this case) supported the centralised configuration.

Centralised testing provides a controlled way of consolidating data, evaluating functional capability, and initiating action. This method can also determine if the product or system is performing as designed - leading to the potential for studying variability in addition to functional capability. Thus, a broader range of analyses can be done from the IoT data to expand testability to the operational performance domain as well.

Considering the financial impact of IoT, it became clear from the centralisation of resources that IoT decreased the overall maintenance and commissioning costs by a factor of three. IoT reduced the number of site visits and labour hours required for any operational or maintenance activities that rely on test data.

A digital agent in the higher-level orchestration layer implements system automation. Such an agent ensures the availability of system status, operational-, performance-, and maintenance data to characterise Pivot systems and to automate test and maintenance functions. The existing functionality can be consolidated to provide a single agent, and adding maintenance control functionality will complete the implementation of such an agent.

Chapter 6

Conclusion

Testability's definition in an IoT system required revision in the context of a full life cycle. IoT widened the scope of testability with the ability to obtain overall system status and to compare that status against the desired status. This definition is based on the fundamental philosophy of testing and focuses on the total system instead of just elements of the system. Action is taken upon any deviation from the desired status. A vital consideration of testability is also to ensure system usability. A system that is considered usable is not necessarily testable, but usability in the form of easily accessible status information and reference data is essential.

A system's status includes information on the functional status of a system, its performance, and information on sub-systems and lower-level system components. This case study's type of information includes configuration, real-time resource conditions, performance, and user activity data. From a full life cycle perspective, the testability of a system becomes a critical design requirement in the design process. The inclusion of testability in the design comes with high connectivity in the operational environment. The reliance on high connectivity makes the system maintenance function's effectiveness highly dependent on the availability of ICT. This dependence must be considered in the design process when conducting failure mode analyses and continuity assurance.

IoT can provide centralised system data of all resources in the IoT system considered in the case study. The concept of a digital agent was developed to ensure each field system is considered an entity with unique characteristics. An agent will monitor and ensure resource availability using all available system data, including data from equipment, field personnel, and environmental data.

From a research perspective, it is necessary to provide a consolidated view of the research. The RVM is used to show the progression of research, to ensure validation is evident, and to demonstrate contributions from this research. Figure 6.1 shows the fully constructed RVM.

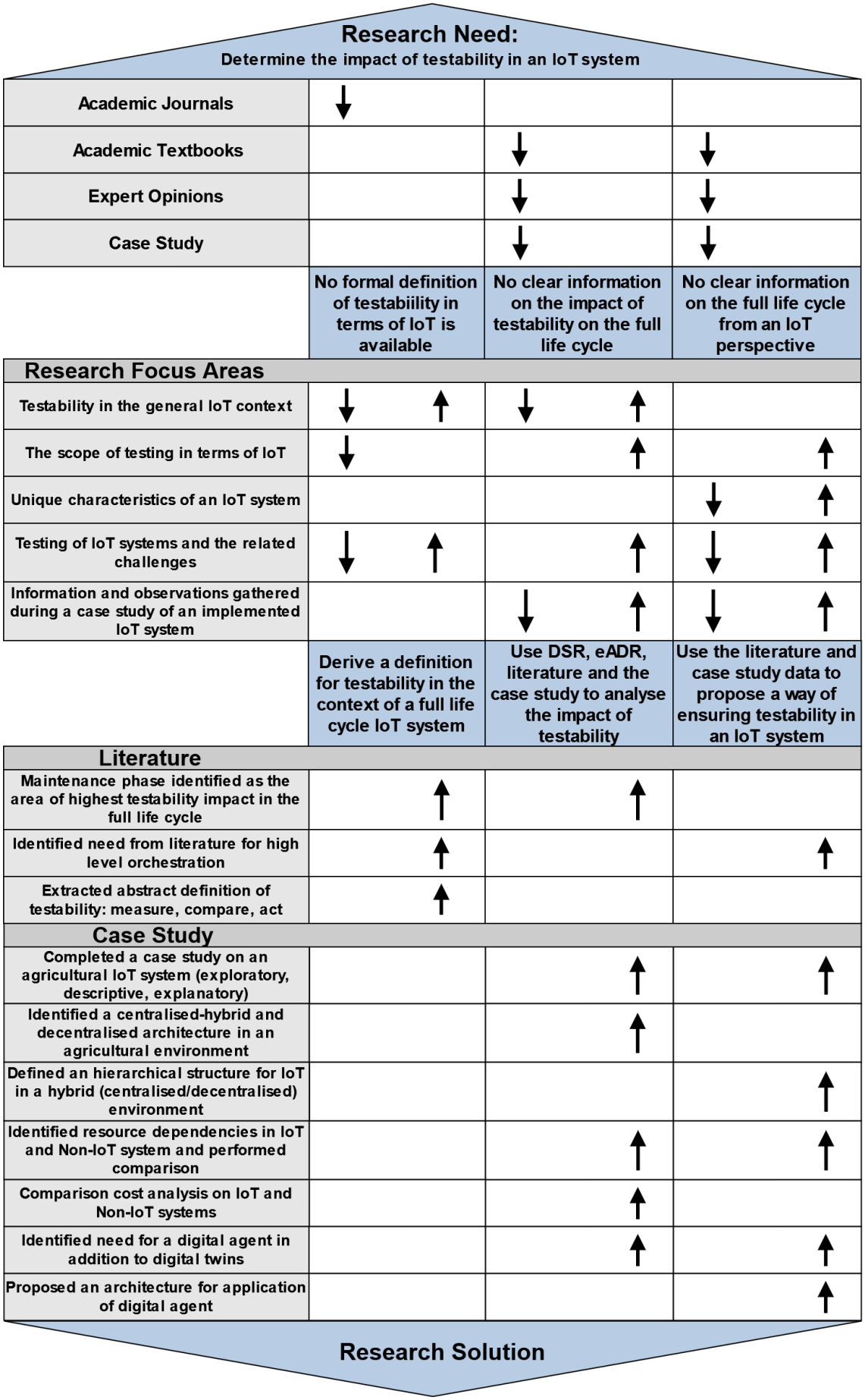

Figure 6.1: Completed RVM of the Study

In figure 6.1, the top section of the RVM validates the research problem. The research problem was initially identified by observing implemented industrial systems and consulting with field experts. The problem was further confirmed by academic research on testability in the IoT context and how testability is applied in the system's full life cycle. The absence of relevant literature on the full life cycle, and the real-world identified need to investigate this topic, validated research into the topic of IoT testability in a system full life cycle context.

The middle section of the RVM in figure 6.1 indicates how literature focus areas aligned with this research to ensure validation of the research problem and to assist in finding solutions to the research challenges. The study first analysed the full system life cycle as defined by SE. This analysis was done to develop an in-depth understanding of the life cycle process defined in the literature. A systematic literature review was then completed to understand how testability is currently incorporated in the full life cycle in the context of IoT.

The literature found that the area in the life cycle where testability would have the greatest impact is in the maintenance phase, where it is necessary to test resources for operational availability and performance. This finding is supported by the results from the case study when the system downtime and the impacting factors were analysed. Analysis of the case study data concentrated on cost and effectiveness (of resource application) and how IoT impacted these factors. It was found that IoT reduced the number of resources required by the system. IoT, in the process, also created a single, high-impact failure point because of the centralised nature of IoT systems, which underlined the importance of good IoT design, implementation, and availability assurance.

The bottom section of the RVM is presented to show how different research activities and findings resulted in a proposed concept for a software artefact (an agent in the high-level orchestration layer) based on the literature study and case study findings. Concept solutions to meet the research challenges are addressed in this section, as discussed:

1. **Derive a definition for testability in the context of a full life cycle IoT system:** From the literature, it was seen that the general definition of testability was not fit to describe the FLC of an IoT enabled system. This was verified in the case study during the analysis of the impact of IoT on the testability of an already field deployed IoT enabled system.

An abstracted definition for testability in the context of a FLC IoT system was obtained by considering the core philosophy of testing: measurement, comparison, and action. In the IoT context, massive measurement can take place remotely using IoT enabled hardware. Comparing the measured values to pre-defined system-wide setpoints can occur automatically or manually, either locally or remotely, depending on the implementation of a digital agent and the IoT architecture. Action can take place remotely (software updates, system reconfiguration) or through a well-informed site visit as IoT made available the system status and provided critical system maintenance data. These factors considered, the revised definition of testability in the IoT context, the ability to determine the status of a system at any point in time, was proposed.

Adopting this definition relied on the existence of a digital agent to consolidate system data, perform comparisons, and initiate action. In the current system (from the case study), data is measured and stored. Still, the consolidation and comparison elements may improve to include continuous analysis of trends and patterns, as can be done in an IoT-enabled system. Without the definition and implementation of a single digital agent implemented as a functional item in software, the testability of the overall system will not be comprehensive.

2. **Use eADR, QRM, DSR, and the case study to analyse the impact of testability:** From the literature, it was identified where IoT would have an impact on the system FLC. The case study identified how IoT will impact the system in different phases, with a focus on areas where IoT has not been fully analysed from a testability perspective. The real impact lies in providing system status based on real-time and historical data on resources' operational and maintenance characteristics.

   Analysis of the case study data revealed that IoT resulted in the centralisation of system resources. Centralisation caused the Pivot system to be less dependent on the system resources but created a single, high-impact point of failure. This indicated the need for highly available IoT resources, including a network, cloud application, and support services. An agent was identified as possible mitigation for the point to monitor and report on the availability of centralised resources.

Without a digital agent, the IoT system would be reduced to a collection of digital twins incapable of orchestrating or automating actions in a combined manner. The addition of an agent thus ensures IoT is applied to the full benefit of the system, not individual components.

3. **Use the case study data to describe the full life cycle in terms of IoT:** The case study found that, to ensure testability in an IoT system, the availability of the system status is required at any point in time on all the layers of the IoT system architecture in the operations and maintenance phases of the life cycle. A reporting functionality must be available, a function that a digital agent must implement in the architecture as it has a consolidated view of the overall system.

   The case study clearly showed the value of centralising resources in an IoT system that reduced cost and dependency on decentralised resources. A clear advantage is evident from the centralisation of SME resources (data analysts) and information, with centralised information also made available to SME field resources (hybrid field application engineers). Field resources can thus be designed, allocated, and managed more effectively.

   A digital agent is thus required in a high-level orchestration layer to fully implement IoT in the full system life cycle.

From the RVM in figure 6.1, research findings can be generalised in the spirit of eADR. From reflection, it is clear that testability must be generalised instead of focusing primarily on development and production testing for IoT systems to reach full potential. In IoT systems, testability has major application in the high-level maintenance task of a full life cycle, with the apparent need to ensure access to testable resources in the system. A high-level orchestration layer must be included to consolidate testing at the correct layer (for the highest impact), accompanied by a specialised digital test agent to handle higher volumes of data. Such an agent will benefit any IoT system as it consolidates measured system status data, compares against individual and collective set points, and initiates appropriate action based on defined operating procedures.

The centralised-hybrid structure in an IoT system can free up subject experts and apply centralised resources more effectively, placing required on-site experts in the hybrid layer and data analysts in the centralised layer. In any IoT system, as is known in systems with a single point of failure, the centralised nature emphasises the availability of the centralised shared resource. These generalisations became evident upon reflection, as confirmed by the case study.

6.1 Future Work

The study scope was limited to investigating testability in maintenance and the role of IoT to reduce downtime. Investigation of uptime can also be considered, and the generalised definition of testability can be applied to ensure operational effectiveness from a quality perspective (measuring performance variability).

The focus of this research was to understand the IoT testability in a larger system, identify shortfalls and potential improvement areas, and define a concept solution. Future research can focus on the design and implementation of a digital agent.

References

[1] J. Engblom, "Internet of things (IoT) in the lab staging and testing for the real world," in *Embedded World 2016*, Nuremberg, Bayern, Germany, Feb. 2016.

[2] M. Bures, T. Černý, and B. Ahmed, "Internet of things: Current challenges in the quality assurance and testing methods," in *Information Science and Applications*, Hong Kong, China, May 2018.

[3] J. Voas, R. Kuhn, and P. Laplante, "Testing IoT Systems," in *2018 IEEE Symp. on Service-Oriented System Engineering (SOSE)*. Bamberg: IEEE, Mar. 2018, pp. 48–52.

[4] G. Muller, "Systems Engineering Research Methods," *Procedia Computer Science*, vol. 16, pp. 1092–1101, 2013.

[5] A. Kossiakoff, W. N. Sweet, S. J. Seymour, and S. M. Biemer, *Systems Engineering: Principles and Practice*, 2nd ed., A. P. Sage, Ed. Hoboken, N.J.: Wiley-Interscience, 2011.

[6] The Australian National University, S. Gregor, A. R. Hevner, and University of South Florida, "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, Feb. 2013.

[7] A. Hevner, "A Three Cycle View of Design Science Research," *Scandinavian Journal of Information Systems*, vol. 19, no. 2, pp. 87–92, 2007.

[8] Hevner, Alan, A. R, S. March, S. T, Park, J. Park, Ram, and Sudha, "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, no. 1, pp. 75–105, Mar. 2004.

[9] J. E. W. Holm and G. P. van der Merwe, "Quality Research Management Improves Design Research Effectiveness," *South African Journal of Industrial Engineering*, vol. 30, no. 3, pp. 238–252, Nov. 2019.

[10] A. M. Petersson and J. Lundberg, "Applying Action Design Research (ADR) to Develop Concept Generation and Selection Methods," *Procedia CIRP*, vol. 50, pp. 222–227, 2016.

[11] M. T. Mullarkey and A. R. Hevner, "An elaborated action design research process model," *European Journal of Information Systems*, vol. 28, no. 1, pp. 6–20, Jan. 2019.

[12] Sein, Henfridsson, Purao, Rossi, and Lindgren, "Action Design Research," *MIS Quarterly*, vol. 35, no. 1, pp. 37–56, 2011.

[13] R. K. Yin, *Case Study Research and Applications: Design and Methods*, sixth edition ed. Los Angeles: SAGE, 2018.

[14] K. Patel, S. Patel, P. Scholar, and C. Salazar, "Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application & future challenges," *International Journal of Engineering Science and Computing,*, vol. 6, no. 5, pp. 6122–6131, May 2016.

[15] P. Ryan and R. Watson, "Research Challenges for the Internet of Things: What Role Can OR Play?" *Systems*, vol. 5, no. 1, p. 24, Mar. 2017.

[16] M. Bures, X. Bellekens, K. Frajtak, and B. S. Ahmed, "A Comprehensive View on Quality Characteristics of the IoT Solutions," in *IoT in Urban Space*. Guimares, Portugal: arXiv, 2018.

[17] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. Aalborg, DK: River Publishers, 2014.

[18] O. Vermesan and P. Friess, Eds., *Internet of Things - from Research and Innovation to Market Deployment*. Aalborg, DK: River Publishers, 2014.

[19] M. Ahmid, "An Agent-Based Approach for the Internet of Things," Ph.D. dissertation, University of Mohamed Khider, Biskra, Algeria, Apr. 2022.

[20] J. M. Samens, "Testability," in *The SAGE Encyclopedia of Communication Research Methods*, M. Allen, Ed. California: SAGE Publications Inc, 2017, vol. 1, pp. 1751–1753.

[21] J. Valfre, "Testability modeling usage in design-for-test and product lifecycle cost reduction," in *2012 IEEE AUTOTESTCON*, Anaheim, CA, USA, Sep. 2012, pp. 39–41.

[22] A. Ahmad, "Model-based testing for IoT systems : Methods and tools," Ph.D. dissertation, Jun. 2018.

[23] "Design for Maintainability," in *Systems Engineering and Analysis*, 5th ed., B. S. Blanchard, Ed. Essex: Pearson, 2014, pp. 476–534.

[24] W. Mengist, T. Soromessa, and G. Legese, "Method for conducting systematic literature review and meta-analysis for environmental science research," *Science of The Total Environment*, vol. 702, p. 134581, Feb. 2020.

[25] PRISMA-P Group, D. Moher, L. Shamseer, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, and L. A. Stewart, "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement," *Systematic Reviews*, vol. 4, no. 1, p. 1, Dec. 2015.

[26] W. Maisiri, "Development of an Industry 4.0 competency maturity model," Ph.D. dissertation, North-West University (NWU), North-West, South-Africa.

[27] I. Popa, I. Lita, and A. Chita, "The concepts, methods and technics for the electronic testability implementation," in *Concurrent Engineering in Electronic Packaging.*, Calimanesti-Caciulata, Romania, 2001, pp. 263–267.

[28] S. Dragos, A. Purcarea, A. Cotorcea, N. Florin, and D. Coșofreț, "Naval maintenance. From corrective maintenance to condition monitoring and IoT. Future trends set by latest IMO amendments and autonomous ships." in *Proceedings of the International Scientific Conference*. Constanta, Romania: MBNA Publishing House, Sep. 2021.

[29] M. Compare, P. Baraldi, and E. Zio, "Challenges to IoT-Enabled Predictive Maintenance for Industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4585–4597, May 2020.

[30] A. Cachada, P. M. Moreira, L. Romero, J. Barbosa, P. Leitno, C. A. Gcraldcs, L. Deusdado, J. Costa, C. Teixeira, J. Teixeira, and A. H. Moreira, "Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. Turin: IEEE, Sep. 2018, pp. 139–146.

[31] R. C. Parpala and R. Iacob, "Application of IoT concept on predictive maintenance of industrial equipment," *MATEC Web of Conferences*, vol. 121, p. 02008, 2017.

[32] A. Khademi, F. Raji, and M. Sadeghi, "IoT Enabled Vibration Monitoring Toward Smart Maintenance," in *2019 3rd International Conference on Internet of Things and Applications (IoT)*. Isfahan, Iran: IEEE, Apr. 2019, pp. 1–6.

[33] J. S. Rahhal and D. Abualnadi, "IOT Based Predictive Maintenance Using LSTM RNN Estimator," in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. Istanbul, Turkey: IEEE, Jun. 2020, pp. 1–5.

[34] Y. Chen, "Integrated and Intelligent Manufacturing: Perspectives and Enablers," *Engineering*, vol. 3, no. 5, pp. 588–595, Oct. 2017.

[35] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[36] T. Bergs, S. Gierlings, T. Auerbach, A. Klink, D. Schraknepper, and T. Augspurger, "The Concept of Digital Twin and Digital Shadow in Manufacturing," *Procedia CIRP*, vol. 101, pp. 81–84, 2021.

[37] N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, no. 2, pp. 277–296, Mar. 2000.

[38] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018.

[39] P. G. Balaji and D. Srinivasan, "An Introduction to Multi-Agent Systems," in *Innovations in Multi-Agent Systems and Applications - 1*, J. Kacprzyk, D. Srinivasan, and L. C. Jain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 310, pp. 1–27.

[40] M. J. Wooldridge, *An Introduction to Multiagent Systems*, 2nd ed. Chichester, U.K: John Wiley & Sons, 2009.

[41] K. Sha, W. Wei, T. Andrew Yang, Z. Wang, and W. Shi, "On security challenges and open issues in Internet of Things," *Future Generation Computer Systems*, vol. 83, pp. 326–337, Jun. 2018.

[42] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10–16, Jun. 2017.

[43] S. Gupta and M. Sinha, "Impact of software testability considerations on software development life cycle," in *Proceedings of 1994 1st International Conference on Software Testing, Reliability and Quality Assurance (STRQA'94)*. New Delhi, India: IEEE, 1994, pp. 105–110.

[44] T.Rajani Devi, "Importance of Testing in Software Development Life Cycle," *International Journal of Scientific & Engineering Research*, vol. 3, no. 5, pp. 1–5, May 2012.

[45] T. Jindal, "Importance of Testing in SDLC," *International Journal of Engineering and Applied Computer Science (IJEACS)*, vol. 1, no. 2, pp. 54–56, Dec. 2016.

[46] D. Richardson, O. O'Malley, and C. Tittle, "Approaches to specification-based testing," *ACM SIGSOFT Software Engineering Notes*, vol. 14, no. 8, pp. 86–96, Dec. 1989.

[47] P. Runeson, Ed., *Case Study Research in Software Engineering: Guidelines and Examples*, 1st ed. Hoboken, N.J: Wiley, 2012.

Appendices

Appendix A

Supporting Documents

A.1 SAIIE 33rd Annual Conference Published Article

TESTABILITY: AN IOT SYSTEM PERSPECTIVE

J.E.W. Holm[1] and J. van Deventer[2*]
[1]Department of Electrical & Electronic Engineering
Management Cybernetics
North-West University, South Africa
johann.holm@nwu.ac.za

[2*]Department of Computer & Electronic Engineering
North-West University, South Africa
contact@jpvandeventer.com
27062686@g.nwu.ac.za

ABSTRACT

The paradigm shift brought about by the Internet of Things (IoT) environment has challenged traditional system analysis and design perspectives. One of the perspectives that has been changed, but may have not received sufficient review, is the way a system and its elements are tested. On the traditional end, testability focuses on test and verification of individual components, their interfaces, and integrated testing in isolated environments. However, the availability of interconnectivity allows testability to cross boundaries, which in turn requires the engineering scientist to revisit the philosophy of testing to highlight the advantages of interconnectivity in system testability. The recent focus of IoT testability has shifted towards software testing using the concept of digital twins, with the assumption that hardware and interconnection components have been tested during development and fabrication. An important challenge that we will address, is thus the concept of an abstract system model that is a real-time reflection of the underlying system state at a point in time. This relies on the availability of the status of all system elements on a continuous basis, where status includes all life cycle information in a multi-agent, multi-disciplinary system. The traditional definition of testability is thus revisited and adjusted to reflect the true intention of testing namely, to acquire the system status in terms of all its elements, and to compare the observed status against a set of desired values that may be defined using a variety of analysis methods over a full system life cycle. A real-world case study and structured literature study are used to show the value of a system perspective on testing in an interconnected (IoT) system.

Keywords: IoT, testability, full life cycle

* Corresponding Author

1 INTRODUCTION

In the past few decades, the development in Industry 4.0 has had large impact on the application of the Internet of Things (IoT) and the Industrial Internet of Things (IIoT) in modern day systems. Testing has become more complex with the development of more advanced IoT systems and networks. This paper analyses the impact of IoT on testability over the full product life cycle.

During the development of IoT systems, a design process is followed that typically has phases including concept, preliminary, and detail design, followed by implementation and testing, and ending in production, operation and maintenance before being phased out (full life cycle [18]). Testability of IoT systems up to this point has proven difficult with no definitive method or mutually agreed upon definition of testing in these types of systems [1],[2],[3].

Testing of IoT systems is typically done to verify the functionality of the system during and after development, during production, at field commissioning, and during operation for performance and availability assurance. In order to revisit the concept of IoT system testability, it is necessary to analyse testing of IoT systems and to learn from analysis, literature, and a case study *where* and *how* the testability of IoT systems is affected.

2 RESEARCH CONTEXTUALIZATION AND METHOD

2.1 Problem definition

The research challenge is to understand the effects of IoT on system testability from a full life cycle perspective, and to propose a method for ensuring testability during operations.

From a real-world perspective, the context is an agricultural irrigation pivot system that has been automated using IoT technology. A number of these systems have been deployed nationally and internationally by the client (owner of the IoT system) for whom this research was conducted.

2.2 Research process / steps

The research process was approached as follows:

1. Understand the impact of IoT on testability (practice-inspired research):
    a. Perform an analysis of testing over a full life cycle from a systems perspective;
    b. Do a systematic literature review on IoT system testability from existing literature; and
    c. Conduct a real-world case study on an irrigation pivot system in the context of agricultural IoT;
2. Reflect and learn from the above impact study (guided emergence):
    a. Identify *where* IoT will add value in the client's system life cycle, with the focus on operation and maintenance;
    b. Define *how* IoT will affect testability by learning from the literature review and case study;
3. Derive a method to ensure testability for the client's real-world system:
    a. Identify areas in the client's maintenance system where IoT will add value from a testability perspective;
    b. Provide a method that will support testability based on the research and bring the method into context.

2.3 Formal research methods

In light of the above process, the following research methods were used to ensure the research obtained a balance between real-world application and theory, as defined in the list below.

1) **Research paradigm and alignment with real world:**
   a) Artefact definition using design science research (DSR) - design science research [11] was used to ensure a recognized paradigm links the real world agricultural IoT system to a grounded theoretical base;
   b) Systems engineering (SE) to identify *where* IoT impacts testability - the client uses systems engineering full life cycle processes for system and product acquisition and utilization [10]. Thus, to align the research with the client process, a full life cycle approach was followed to analyse the areas where IoT testability would be present in the life cycle;
   c) Alignment using quality research management (QRM) - quality research management [12] provided structure to align client requirements and research outputs by listening to "the voice of the customer". This was done inside the DSR paradigm and provided goal-driven research challenges and solutions;
2) **Case study research:**
   a) Case study research to understand how IoT impacts testability - due to the complexity of IoT systems, and since the boundaries between the abstract (logical) representation of the system and the real-life context are not clear, it was necessary to use a case study. Observations from case studies were used to indicate *where* and *how* IoT principles will affect testability [24];
   b) Elaborated Action Design Research (eADR) to support action research - elaborated action design research [13],[14],[15] allowed real-world activities to inform the case study in a structured manner. The case study allowed analysis and reflection of the implementation and evolution eADR cycles with an artefact as outcome (the resulting method derived from findings).

3 LITERATURE STUDY

3.1 Testability in the systems engineering full life cycle

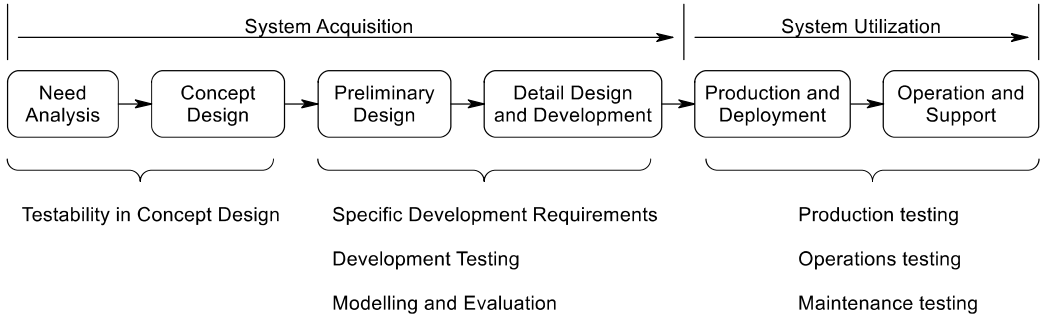

Figure 1: Testability in the Full Life Cycle [18]

From the above process [18], it is evident that testability forms part of the overall life cycle, and it is thus important to understand the impact of testability downstream in order to specify correctly in the first phases above. With production testing mostly contained and defined in a controlled environment, the focus of this research shifts to the high-level Operation and Support where testing is used in the utilization life cycle phase. Requirements, design

references, model evaluation, and detail system testing are all verified in the systems engineering process, which is also a form of testing.

Specifically, testing in the Operation and Support high-level task is important since a real-time view of the overall system status is important for timely failure monitoring, reactive and preventive maintenance, and system performance management.

3.2 Testing during maintenance and operation

The corrective maintenance cycle can be reduced to six phases as indicated in Figure 2 [18].

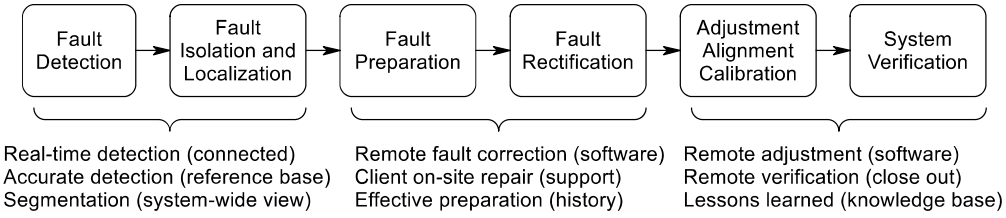

Figure 2: Corrective Maintenance and IoT-Related Activities (adapted from [18])

In terms of testability, the IoT system will provide the ability to:

1. Perform fault detection either reactively (functional capability failure) or proactively (deviation from expected value) using remotely obtained system status data. Historical data will be available to (i) identify a predefined fault (pattern recognition), or (ii) to report on exceptions (anomaly detection);
2. Support fault isolation from a system view (as opposed to a limited view) by using data from nodes and interconnections. Historical data from system logs will support this effort as actionable information;
3. Preparation for repair can be done using historical data to derive system characteristics. Failure types, inferred from IoT status data, will prepare the workforce for a site visit;
4. Fault rectification can be done either remotely, by a person on the client site, or by means of a dedicated site visit from a service technician. IoT will enable and support the prior two options;
5. Adjustment may be done reactively or proactively based on real-time system status data. Verification is done in a similar manner.

When testability is considered as the ability to obtain system status data and to compare this data against a valid reference, it becomes possible to perform performance (functional capability) testing and condition (resource integrity) testing.

Without IoT, a minimum of 2 site visits will be required. At the first visit the fault would need to be identified (testing), and required tools and components be acquired, followed by a second site visit. If unsuccessful, a third visit may be required. With IoT, a single planned site visit may be sufficient using historical and real-time status data.

The same argument holds for preventive maintenance, where the availability of system status will support workforce management (effort and cost) by having available relevant system status data.

Similarly, performance testing (doing installation and operating the system) is equally supported by having performance data available for quality management purposes. That is, to obtain relevant, real-time performance data for comparison with valid desired performance values.

3.3 Testability in an IoT system

Testability is the extent to which a system or unit supports fault detection and fault isolation in a confident, timely and cost-effective manner [9]. In general, testability is the ability to run an experiment to test a hypothesis or theory [8]. Thus, testability implies the ability to (i) define the desired state of a system, (ii) obtain the status of a system, (iii) compare the obtained status against the desired status, and (iv) make available actionable information.

From the research, the scope of testing was limited to device testing – a system view was not provided, as such.

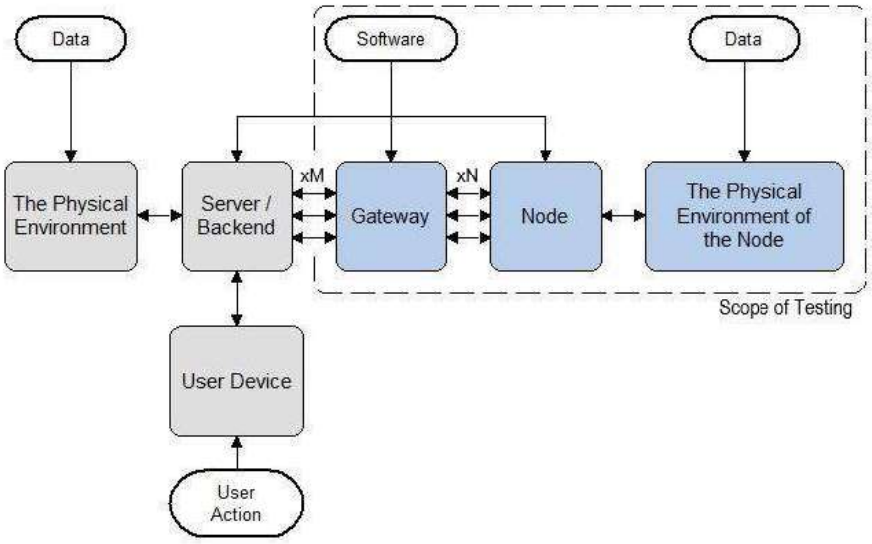

Figure 3: Scope of Testability in Terms of IoT (adapted from [1])

Full testing in an IoT system is similar to fully testing large sections of the internet. While the possibility of fully testing the internet is unlikely, it is sensible to create sub-networks within the greater IoT network. Sub-networks, defined as Networks-of-Things, are bounded components of the IoT network that support realistic testing [3].

A question of interest concerns how the size of the network affects testability. With large-scale networks, large amounts of data will be produced. The data, after processing, is likely to be used in making decisions and taking different actions [3].

Physically, a large IoT network will rely heavily on simulations when testing sub-systems. Large networks will by definition have larger numbers of nodes that cannot be all individually tested. With the difficulties in the testing of smaller-scale IoT networks already discussed, the reliance of large IoT networks on well-modelled simulations for hardware testing is clear.

3.4 Systematic literature review

As part of the research process, a systematic literature review was done on IoT system testability [16],[17]. Inclusion and exclusion criteria are provided in Table 1 below:

Table 1: Criteria for source inclusion or exclusion

	Criteria (Code)	Criteria Description
Inclusion	Closely Related (CR)	The text relates to general testability or testing of IoT systems
	Application Relevance (AR)	The application type used for IoT, or testability provides insight into the research problem
Exclusion	Duplication (DP)	The text appears multiple times within the same criterion
	Language Compatibility (LP)	The full text is not accessible in English
	Text Length (TL)	The full text is not available or accessible
	Context (C)	The context wherein IoT or testability is used differs too much from the context of the study
	Casual Use (CU)	The terms IoT and testability are not discussed or investigated in depth to provide relevant information

3.4.1 Text selection process

The text selection followed a four-stage process with two exclusion stages as indicated in Figure 4. In the first exclusion stage, the texts were removed if found to be duplicates (DP), not available in full English (LP), or the full length of the text was not available (TL). Having passed the first exclusion stage, the texts were removed in the second exclusion stage if the research topic was found to be in a non-applicable context (C), or the text used the terms testability and IoT in a casual way (CU) without providing any in-depth definitions or discussions on the topic.

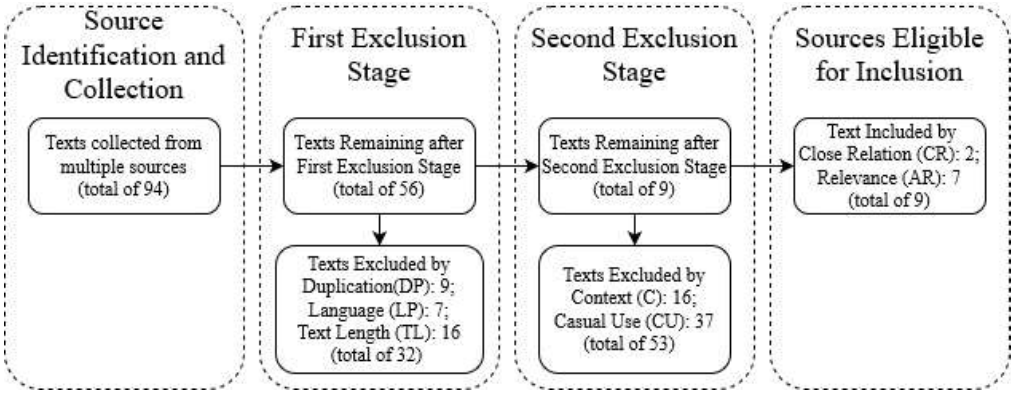

Figure 4: Four-Stage Systematic Text Selection Process (adapted from [25])

Four main focus areas emerged from the search, namely:

- Testing of IoT systems and related test challenges;
- Scope of testing in terms of IoT;
- Unique characteristics of an IoT system;
- Testability in the general and IoT context.

The limited scope of IoT testability was confirmed from the above (mainly considering testing within the scope of Figure 3, the device itself). There was no literature specifically on the impact of IoT from a system full life cycle perspective as shown in Figure 1. It was found that there is limited in-depth research specifically on testability in the IoT space, and how it differs from conventional system testing.

The following was found from the systematic literature review:

- Testing of IoT systems and related test challenges [1],[2],[3] – A significant challenge is the size of IoT networks [1],[3]. Each node on such an IoT network is linked to multiple sub-systems/nodes to handle communication, control, data capturing, and processing. The status of each node must be available, as well as communication links between nodes. Critical communications links were not specifically addressed in the literature that was reviewed, but must be considered in real-world systems (for failure mode effect analysis);
- Scope of testing in terms of IoT [1],[3] - The scope of testing in an IoT environment is limited mostly to functional testing of devices and their features. This limited perspective means that testability will not take a full life cycle system view (a product as opposed to a system view), which will result in less-than-optimal system status information being available for life cycle intelligence;
- Unique characteristics of an IoT system [4],[5],[6],[7] - An IoT system is characterized as a large network of multiple different hardware, software, and human elements implemented with the goal of generating, collecting, and processing data. Characteristics of IoT systems include the following, namely: interconnectivity (massive); heterogeneity; dynamic nature; enormous scale; and data security;
- Testability in the general and IoT context [8],[9] - Testability in the general context is mostly limited to test processes, procedures and methods in detail design, production testing, and diagnostic testing (in general) in the system full life cycle. Reference to full life cycle testing was not found in the review. This limited view prevents operations testing issues from being used as cases in the system acquisition phase.

In summary, the concept of testability as a system life cycle consideration was not explicitly found in the literature review. Test methods were limited in scope and mostly to functional testing of a node or "thing". In general, the definition of testability relies heavily on the concepts of controllability and observability testing as stated in [9],[22],[23],[24],[25]. It can therefore be concluded that testability can be described as the ability to establish the status of a system and to compare that status against a desired reference state.

3.5 The concept of an IoT agent in a multilayer system

In a general IoT system, the concept of a multilayer system as well as the concept of an agent are important as these form part of the testability analysis and the definition of a method (or framework) to be provided [23]. The layers include:

1. Layer 1 - Perception layer: Devices, sensors and actuators are present. In the case of the client's agricultural IoT system, this layer includes a controller connected to local devices via local wireless and copper networks;
2. Layer 2 - Network layer: Short-range, long-range cellular and fibre networks are typically used, with a proprietary ad-hoc local wireless network, copper network, and a cellular gateway being the main components at client sites;
3. Layer 3 - Middleware layer: This cloud-based layer is used for management of devices, connectivity, data, and service. In case of the client site, this layer is important as this is where testability will be ensured (as a quality management function);
4. Layer 4 - Application layer: Consumer and business applications appear in this layer. In the client's case, this layer includes web applications for smartphones and computers, as well as business applications for management of enterprise farm systems.

An agent is defined as a physical or virtual entity that is sufficiently resourced, and being (i) autonomous, (ii) flexible in terms of being reactive, proactive, and socially connected, (iii) situationally aware, (iv) capable of reasoning, and (v) adaptive [23].

All the characteristics above are not always present in an agent, depending on the need. At the onset of this research study, the presence of agents (as such) in higher layers did not exist. The client system had need for a method to ensure testability, with an agent being a candidate option to implement a method to ensure testability. The case for both proactive and reactive approaches is made as historical and real-time data is available.

4 CASE STUDY

The case study was used to obtain insight into the impact of testability in the real world. Observations from the case study were used to augment the findings from the analysis and the systematic review.

4.1 Case study design

This is an exploratory study to answer the questions: “where”, “how”, and “how much” IoT capability will impact the overall system testability [24]. The focus is on the utilization phase of the irrigation system, with specific focus on scheduled and unscheduled maintenance and general system support.

The proposition is to use this case study in the Operation and Support task to evaluate the effects of IoT capability on test effort and cost. System repair may take place either remotely (software or configuration) or on-site (hardware or irrecoverable software failure).

The units of analysis will thus be work effort and associated cost (for example, site visit travel cost). The outcome of effective testing is thus that the system down-time and cost of testing are reduced.

A representation of an agricultural irrigation pivot point system and sub-systems of this case study is provided in Figure 5 (below). Note that central to the system is an IoT Master unit that acts as a gateway. In this system, the water supply, electrical supply, and weather sub-systems are connected to the IoT Master by means of a local ad hoc wireless network. The interfaces between the Master and sub-systems provide additional information to a local digital agent.

4.2 Participants

The research team comprised the following participants, as outlined in the list below.

- Primary researcher: Participant-observer doing (pre-)installation testing, testing as part of corrective maintenance, and recording and analysis of observations;
- Research leader: Observer assisting with data analysis and case study design;
- Technical team: Client team doing on-site installation, testing, and system status monitoring during operations and for maintenance tasks;
- Cloud development team: Client team doing development in the Middleware layer (layer 3);
- End-user team: Farmer (or representative) performing inspection, on-site repair, and minor service.

SAIIE **4.3 Case study system context**

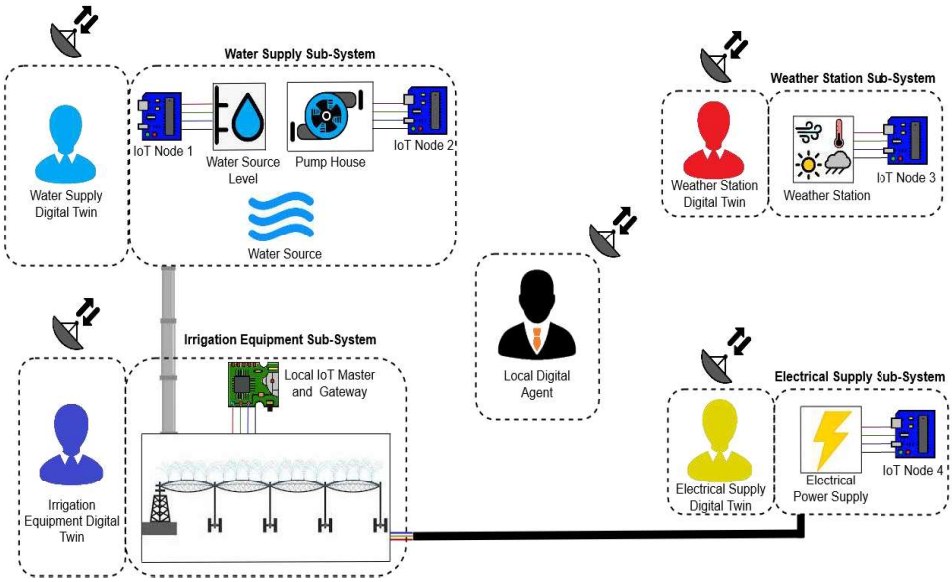

Figure 5: Irrigation Pivot IoT Field System Architecture

4.4 Data collection

Data collected in the study was in the form of observations during the installation, operation, and maintenance phases of the life cycle (that is, the utilization phase). The intrinsic nature of the data resulted in some system elements being less informative.

Specifically, the following process was followed for test analysis:

- For each system task, observations relating to testing were made and recorded. These tasks include installation and repair testing;
- System status measurement (IoT) was observed and noted. For example, 3-phase power presence, fluctuation, and failure;
- Root-cause analyses of underlying faults were done. Observations were linked to measured data to find the role of IoT functionality in the test effort;
- Failure modes were identified, and associated cause-effect analysis provided information on the actual failure, its diagnostic data, and the impact of each failure;
- Operations data was also obtained to describe the effects of human-system interaction on site, as well as measurements relating to sub-system performance;
- Directly measured data on system status was used to report and rectify faults, while in other cases, inferences were made from measurements, their dynamics, and their trends.

4.5 Case study results and analysis

In the data collection phase, the impact that IoT has on such a system was identified on an operational level. A comparison was made for the field system with and without on-site IoT equipment. A summary of incident types and their root causes is presented below:

Primary failures

- Electrical supply failure: Incorrect wiring (phases not connected, phase sequences incorrect), Eskom supply failure, load-shedding, power fluctuation, short circuits,

lightning strikes. Observed using IoT status on power line monitoring that required a site visit. Loss of system communication was due to a lightning strike. Required site visits by an electrical contractor;

- Water system failure: Incorrect system design, pump system failure, valve failure, blockages in the water line, pressure drops due to losses. Observed using power flow and pressure monitoring. Required extensive system audit (design), pump equipment replacement (failure), and on-site removal of obstructions. Obstructions could be removed by the end-user without a dedicated site visit;
- Pivot structure failure: Tower motor drive failure, safety system failure, wet condition failures (wheels losing traction), and structural failure. Observed using dynamic anomalies from measuring rotation angle, speed, and safety signals. Could be rectified using end-user team without a dedicated site visit;
- Pivot controller failure: Configuration error, equipment module failure, communication failure (not a critical failure by design), and software failures. Observed using system configuration and operational status monitoring, and event logs. Configuration changes did not require site visits, but equipment failure required a dedicated site visit;
- Communication failure: In the Network layer, failure to communicate, slow speed on network, and power failure. Observed using network layer monitoring software. On-site power failures resulted in communication failure, while slow networks were characterised, and changes were made to antennas based on link quality. Required site visits for installation failures but reduced future visits upon fault rectification;
- Operator failure: End-user errors, system tamper, and incorrect system configuration. Observed using Middleware software and Application layer software to detect system status change and consequent failures. Could be resolved remotely by providing client support;
- System performance: Irrigation performance, total operating hours (up-time) and down-time, and failure frequency. Observed using operational performance status monitoring and comparing against desired set values. Did not require site visits for support, but notifications could be provided to end users;
- Configuration and reconfiguration: After installation or after fault rectification, the system could be reconfigured remotely without a site visit specifically to this end. Configurations could be verified against historical configurations or defined set values as a form of testing.

The inclusion of IoT thus, in general, increased the efficacy of corrective maintenance by reducing the number of site visits typically to a single visit in the case of a hardware fault, and none in the case of a recoverable software fault. IoT functionality decreased the total system down-time and time required to detect, isolate, and prepare for fault rectification.

IoT provided system performance and technical logs that supported preparation for scheduled maintenance as all performance data was available in the Middleware layer before a site visit. Performance data could be used to identify anomalies, while technical logs provided maintenance history.

4.6 Cost and time impact

The case study was conducted over a limited period of time (1 year), and time and cost savings can be illustrated for single visits. Distances to sites varied significantly, from 50km to 400km from the depot, hence the cost saving becomes a major consideration for remote sites. In a single visit, the time and cost spent on the road as well as on-site work effort (cost and time) can be estimated fairly accurately based on actual site visits.

The following values were used for time and cost estimation (for a specific site close to the depot – for simplicity, set at 50km):

- Fuel price: R 21.61 (at the time of the study)
- Average fuel consumption of vehicle: 10-15 km/l
- Labour fees per technician: R 105 per hour (two technicians per site visit)
- Average duration of site visit: 6 hours
- Travel time: 1 hour
- Distance to site: 50 km
- Cost of single site visit: R 1,509.06 – R 1,581.10

For a site located at 400km, the following applies:

- Average duration of site visit: 6 hours
- Travel time: 10 hours
- Overnight cost: R 850 (total for both technicians)
- Distance to site: 400 km
- Cost of single site visit: R 4,312.52 – R 4,888.80

The cost and time savings per annum was estimated for sites located 50km and 400km from the depot. From site visits in 2021, the following average estimates can be made:

- Site visits per annum without IoT: 6 visits including fault isolation and delays
- Site visits per annum with IoT: 2 where planning is done in advance
- Mechanical lifetime of mechanical structure: 20 years

With the above values, it can be estimated that the system maintenance costs would be as follows over the system lifetime:

- 50km from depot:
    - Without IoT enabled, the cost would be R 181,087 – R 189,732. In comparison, with IoT the cost would be R 60,362 – R 63,244;
    - Time spent on repairs for a site close to the depot would be 35 person-days without IoT, and 12 with IoT.
- 400km from depot:
    - Without IoT enabled, the cost would be R 517,502.40 – R 586,656 in 2021 terms. An IoT enabled system would be R 172,500.80 – R 195,552 for a system with IoT;
    - The time spent on repairs would be 80 person-days without IoT, and 27 days with IoT.

Production down-time and other losses were not considered. Also, costs will increase significantly as travel costs (cost to service provider) will increase. Travel costs to the end-user will be at a higher rate, and repeat visits were not considered (wastage and spillage were not considered).

The saving is thus a factor 3 based on site visit statistics from 2021. This is an estimate as sites often required more on-site work not directly related to testing, and an estimation was made for work done specifically on testing.

It is thus evident "where" IoT testability affects the system life cycle most, namely in the Operation and Support phase of the life cycle. More specifically, testing is mostly done in corrective and preventive maintenance tasks, although performance is also measured and verified.

4.7 Reflection

Based on the system life cycle analysis, systematic literature review, and the case study results, a generalized conclusion can be drawn as shown in Table 2:

Table 2: "How" of IoT on testability

With IoT Implemented	Physical Action	With IoT Not Implemented
Requires no site visits unless the system is offline. Data can be retrieved remotely at any point in time at very low cost.	***Long-term monitoring***	Requires multiple scheduled site visits. Requires hardware capable of storing data until the following data retrieval.
Site visits are reduced unless the system is offline. Faults are identified and restored remotely, or with a single site visit. Time on site is reduced as the fault is known.	***Fault identification and restoration***	Requires site visit. Multiple site visits may be required if the identified fault requires specialised equipment to be restored.
Requires no site visits unless fault cannot be detected or reported by the system. The system can automatically report any faults and notify the relevant parties. This also results in minimal system downtime.	***Fault detection***	Impossible to detect possible fault immediately. Fault detection done after the system has halted operation. Site visit is required in some cases
Requires no site visit unless an in-depth reconfiguration is needed, or the system is offline. The system can be configured in any way remotely at any time at low cost.	***System configuration***	Requires site visit at every configuration. Configurations can occur at any point in time depending on the site. Changes in environmental conditions, for example, could require a system reconfiguration.
Requires no site visit or physical human interaction with the equipment unless the system is offline. The user can fully access, control and monitor the system remotely.	***User operation***	Monitoring and control are only possible when on site, which requires travel and time. No status available remotely, leading to losses and effort.

It is evident that the availability of a reliable network is critical, but also that a system must be designed to function without such a connection. In such instances, testability will be affected as system status data will be delayed.

5 ENSURING TESTABILITY

To ensure testability, it is necessary to measure system status and performance in different layers. From a logical perspective, a digital agent may be employed to acquire system status and performance data from different sources [23]. Its main purpose is to monitor system element status and operational performance, to compare status and performance data against desired set values and provide actionable information as an output. Figure 6 indicates the application space of the digital agent within the context of this study, whereas Figure 7 shows the functional structure of the agent.

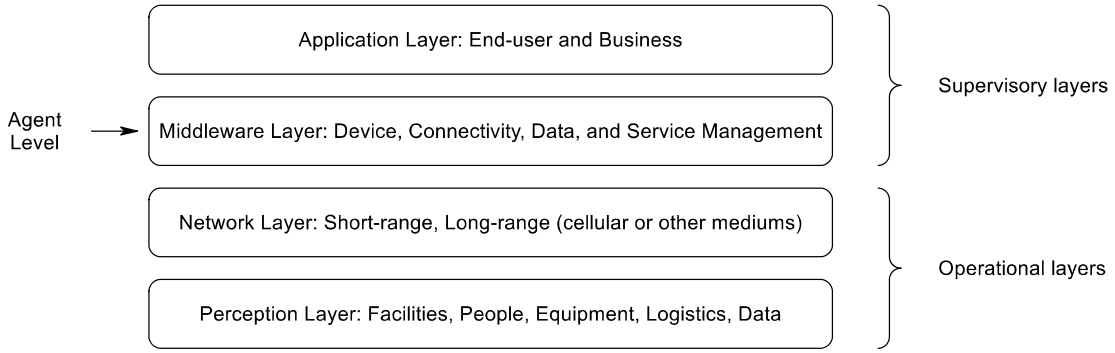

Figure 6: Digital Agent in the Middleware Layer

A Digital Test Agent is thus introduced as a software implementation of an autonomous entity to ensure focus on testability. The importance of testability in an IoT system has been identified from the study and the application of a Digital Test Agent will allow a single entity to charaterize a complete installation on site, with all its metadata, historical data, desired and expected reference values, with actionable information (intelligence) as output to the system Middleware and Application (management) layers.

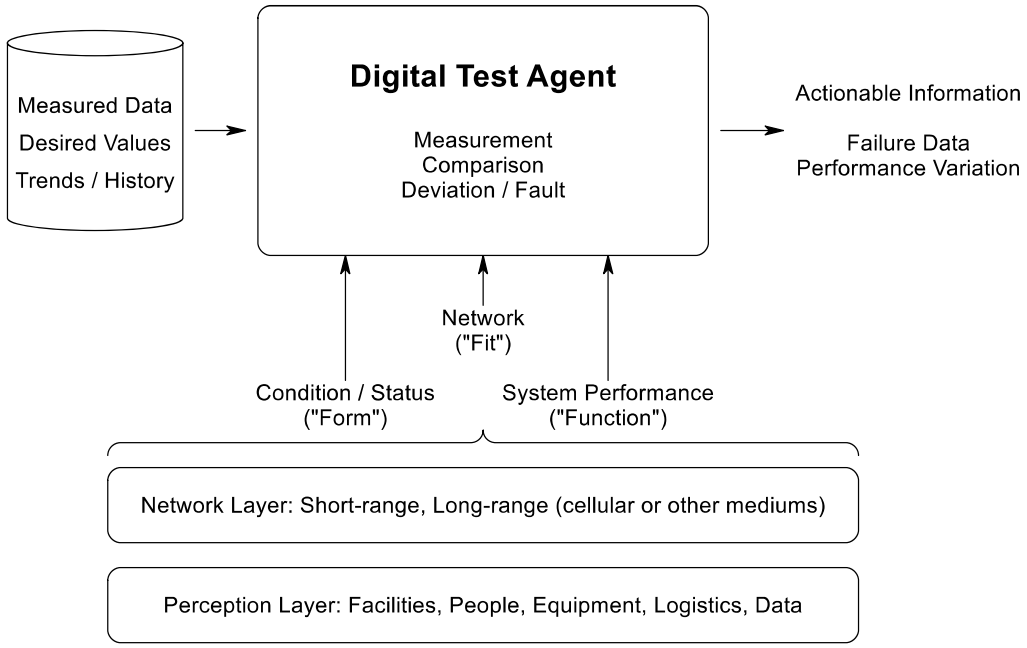

Figure 7: The Structure of a Digital Test Agent

6 CONCLUSION

It is concluded that testability, in this context, is the ability to establish the performance and resource condition status of a system, and to compare that status against a desired reference state or expected trend.

Usability becomes important when considering what a testable system looks like. In basic terms, a system that is usable is not always testable, but usability in the form of "ease of access" to status information is important.

From a full life cycle perspective, a system's testability is designed by assuming connectivity and the critical dependency on an available network. This must be stressed as the system's overall performance will become increasingly dependent on data availability. In the case study, the system was designed to also function without access to data (with limited efficacy).

IoT supports remote status data acquisition in real-time, resulting in significant advantages in system characterisation, availability assurance, and workforce management and planning. Effort and costs are saved, and centralised monitoring and control are utilised to ensure system operational availability. Remote access to data provided an opportunity to perform timely fault and variance detection since most processing was done centrally. This type of status data would not be available without a reliable IoT network, underlining the need for a robust, stable IoT network.

In the case study, remote availability of system status data and metadata reduced site visits for both the end user and technical staff, reducing lifetime cost by a factor of around 3. Operational loss reduction is achieved from timely and focused repairs but may be as much as the loss of a total production unit due to lack of irrigation.

The concept of a Digital Agent was identified to ensure testability is supported by measuring system performance and status data. Such an agent will be present in the Middleware layer and will focus on an installation as a whole. Its purpose is to measure status and performance data, compare against trends and desired values, and provide actionable information.

The IoT system is heavily reliant on interconnectivity, which remains a challenge in an environment with less-than-optimal available resources (failing electricity), but its value is evident from the analysis and case study.

Future work will include analysis of additional case study analyses and refinement of the concept to include advanced intelligence.

7 REFERENCES

[1] J. Engblom. (Feb. 2016). Internet of things (IoT) in the lab staging and testing for the real world. Presented at Embedded World Conference '16 [Online]. Available: https://www.researchgate.net/publication/301230219

[2] M. Bures, T. ˇCern´y, and B. Ahmed, "Internet of things: Current challenges in the quality assurance and testing methods," presented at 9th iCatse Conference on Information Science and Applications, Hong Kong, China, 2018.

[3] J. Voas, R. Kuhn, and P. Laplante, "Testing IoT Systems," in *Proc.*2018 IEEE Symposium on Service-Oriented System Engineering (SOSE). Bamberg: IEEE, Mar. 2018, pp. 48–52, doi: 10.1109/SOSE.2018.00015.

[4] K. Patel, S. Patel, P. Scholar, and C. Salazar, "Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application & future challenges," *International Journal of Engineering Science and Computing*, vol. 6, no. 5, May, pp. 6122-6131, 2016.

[5] M. Bures, X. Bellekens, K. Frajt´ak, and B. Ahmed. (Dec. 2018). A comprehensive view on quality characteristics of the IoT solutions. Presented at 3rd EAI International Conference., Guimares, Portugal. [Online]. Available: https://arxiv.org/abs.182.09683/

[6] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. Aalborg, DK: River Publishers, 2014.

[7] O. Vermesan and P. Friess, Eds., *Internet of Things - from Research and Innovation to Market Deployment*. Aalborg, DK: River Publishers, 2014.

[8] J.M. Samens, "Testability," in *The SAGE Encyclopedia of Communication Research Methods*, Vol. 1, M. Allen, Ed. California: SAGE Publications Inc, 2017, pp. 1751-1753.

[9] J. Valfre, "Testability modeling usage in design-for-test and product lifecycle cost reduction," in *2012 IEEE AUTOTESTCON*. Anaheim, CA, USA, Sep. 2012, pp. 39-41.

[10] S.M. Biemer, A. Kossiakoff, S.J. Seymour, and W.N. Sweet, *Systems Engineering: Principles and Practice,* A.P. Sage, Ed., Hoboken, N.J., USA: Wiley-Interscience, 2011.

[11] S. Gregor, and A. R. Hevner, "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly*, vol. 37, no. 2, Feb., pp. 337–355, 2013.

[12] J. E. W. Holm and G. P. van der Merwe, "QUALITY RESEARCH MANAGEMENT IMPROVES DESIGN RESEARCH EFFECTIVENESS," *South African Journal of Industrial Engineering*, vol. 30, no. 3, Nov., pp. 238-252, 2019.

[13] A. M. Petersson and J. Lundberg, "Applying Action Design Research (ADR) to Develop Concept Generation and Selection Methods," *Procedia CIRP*, vol. 50, pp. 222–227, 2016.

[14] Sein, Henfridsson, Purao, Rossi, and Lindgren, "Action Design Research," MIS Quarterly, vol. 35, no. 1, Mar., pp. 37-56, 2011.

[15] A. R. Hevner and M. T. Mullarkey, "An elaborated action design research process model," *European Journal of Information Systems*, vol. 28, no. 1, Jan., pp. 6–20, 2019.

[16] W. Mengist, T. Soromessa, and G. Legese, "Ecosystem services research in mountainous regions: A systematic literature review on current knowledge and research gaps," *Science of The Total Environment*, vol. 702, Feb., pp. 1-11, 2020.

[17] PRISMA-P Group, D. Moher, L. Shamseer, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, and L. A. Stewart, "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement," *Systematic Reviews,* vol. 4, no. 1, Dec., p. 1-9, 2015.

[18] B. S. Blanchard, Ed., *Systems Engineering and Analysis,* 5th ed. Essex: Pearson, 2014.

[19] S. Gupta and M. Sinha, "Impact of software testability considerations on software development life cycle," in Proc. 1994 1st International Conference on Software Testing, Reliability and Quality Assurance (STRQA'94). New Delhi, India: IEEE, 1994, pp. 105–110.

[20] T.Rajani Devi, "Importance of Testing in Software Development Life Cycle", *International Journal of Scientific & Engineering Research*, vol. 3, no. 5, May, p. 1-5, 2012.

[21] Tanu Jindal, "Importance of Testing in SDLC", *International Journal of Engineering and Applied Computer Science (IJEACS)*, vol. 1, no. 2, Dec., p. 54-56, 2016.

[22] I. Popa, I. Lita, and A. Chita, "The concepts, methods and technics for the electronic testability implementation", in *24th International Spring Seminar on Electronics Technology*. Calimanesti-Caciulata, Romania, 2001, pp. 263–267.

[23] Ahmid, M, "An Agent-Based Approach for the Internet of Things", PhD Thesis, Mohamed Kihder University, Biskra, ALGR, 2021.

[24] Yin, R.K,. *Case Study Research Design and Methods*, 5th ed. CA, USA: Sage Publications, 2014.

[25] Maisiri, W., "Development of an Industry 4.0 competency maturity model", PhD Thesis, North-West University,Potchefstroom, SA, 2022.