Full life cycle analysis of usability in IoT systems

E du Plessis

orcid.org/ 0000-0002-8777-5399

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: May 2020

Student number: 24059706

ACKNOWLEDGEMENTS

Firstly, I would like to thank God for this opportunity and for helping me every step of the way. God gave me the faith, wisdom, patience, and endurance needed to finish this study.

I would also like to thank Prof Holm for all the wisdom, knowledge, patience, and time he gave to this study. I appreciate everything you did for the sake of this study. Supervising a masters study is no easy task, and you are truly an excellent supervisor. Also, to the family of Prof Holm, thank you for your patience and understanding through these two years.

To Jericho Systems, thank you for the time, knowledge, facilities, and financial support you provided during this study.

I would also like to thank the North-West University for their contribution, including the use of their facilities.

To my parents and sister, thank you for your support, patience, love, and understanding through these two years.

To my boyfriend, Leroux, thank you for all the love, support, patience, assistance, understanding, and care you gave me through this study. Even through the tough times, you were always there.

ABSTRACT

Usability in IoT systems has not been clearly defined, and even more so over a system full life cycle. The concept of usabililty over a full life cycle is not commonly encountered, and a research question arose as to its definition. A comprehensive literature study validated the limited definition of usability in IoT system life cycle phases. Literature also supported the observation that usability has its main focus on end-users during the system's operational and maintenance phase. A usability baseline is thus needed to define usability over all system life cycle phases and must include all system users and stakeholders from a system perspective.

Firstly, a definition for general IoT systems was derived to form a foundation for the usability framework that includes all system life cycle phases, users, and stakeholders. Secondly, the life cycle usability framework was developed using Nielsen's usability heuristics as a baseline from which to develop a set of generalised usability heuristics that can be applied to IoT systems, as opposed to Nielsen's end-product usability view.

The framework was validated by applying the life cycle heuristics to general usability issues in IoT systems as obtained from literature, a peer-reviewed IEEE conference article that commended the work, and by application of the life cycle heuristics to the development and successful deployment of a centre pivot irrigation system (CPIS). The life cycle usability heuristics were found to address the general usability issues, as well as improving the perspective and definition of usability over the life cycle of the CPIS.

Many of the life cycle usability heuristics were found to be addressed by systems engineering functions, with model-based systems engineering adding notable value. The value of systems engineering showed that proper application of systems engineering processes and methods, augmented with effective contextualisation, constructivism, complexity reduction, and effective communication form a valid full life cycle usability baseline for IoT systems.

KEYWORDS

Usability, system, full life cycle, heuristics, centre pivot irrigation system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADR	Action Design Research
ASQ	After Scenario Questionnaire
BIE	Build, Intervention and Evaluation
CP	Centre Pivot
CPIS	Centre Pivot Irrigation System
C-PLM	Closed-loop Product Lifecycle Management
CPS	Cyber-Physical System
CRM	Customer Relationship Management
DFU	Design for Usability
DIT	Design, Implementation, and Testing
DM	Design Management
DSR	Design Science Research
DSRM	Design Science Research Methodology
eADR	Elaborated Action Design Research
E/HF	Ergonomics and Human Factor
ERP	Enterprise Resource Planning
FMEA	Failure mode and effect analysis
FMECA	Failure mode, effects and criticality analysis
HFE	Human-factor Engineering
INCOSE	International Council on Systems Engineering
IoE	Internet-of-Everything
IoT	Internet-of-Things
IIoT	Industrial Internet-of-Things
IS	Information system
IT	Information technology
LED	Light-emitting Diode
LEPA	Low Energy Precision Application
MBSE	Model-based System Engineering
O&M	Operations and Maintenance
PUEU	Perceived Usefulness and Ease of Use
PR	Product Retirement
QUIS	Questionnaire for User Interface Satisfaction
RA	Requirements Analysis
RAMI4.0	Reference Model Industrie 4.0
RE	Requirements Engineering

RFID	Radio-frequency Identification
RM	Requirements Management
PR	Product Retirement
SEMP	Systems Engineering Management Plan
TEMP	Test and Evaluation Management Plan
UI	User Interface
URS	User Requirements Specification
WBS	Work Breakdown Structure
WSM	Warehouse Stock Management
WYSIWYG	What You See Is What You Get

1 Introduction

Usability in Internet-of-things (IoT) systems lacks in the engineering industry. Issues such as users misinterpreting information, due to a lack of context or simply not having access to the necessary information reflect this usability insufficiency. Usability is undervalued and underestimated, and this study aims to shed some light on how the efficient implementation of usability can improve IoT systems, not just regarding the product but also the overall user-experience.

To validate the issue of usability and in the spirit of action design research, research was done on usability in IoT systems through an extensive literature study. Firstly, it was found that there is no clear definition of IoT systems in general. Secondly, there is a general lack of usability in IoT systems and the full system life cycle. Lastly, where usability is currently applied, the focus is mostly on the end-user.

The proposed research solution is the development of a usability framework in IoT systems over the full system life cycle. The purpose of the framework is to provide a baseline for increasing usability in full system life cycles. Part of the framework is a definition of IoT systems in general, where all users and stakeholders in all life cycle phases are included. During the literature study, usability heuristics were identified as the appropriate choice of usability testing and Nielsen's usability heuristics were identified as the most complete set of usability heuristics. As Nielsen's usability heuristics are applied to products, a generalised set of usability heuristics were developed for system usability.

The framework is validated with a peer-reviewed IEEE article, applying the framework to general usability issues in IoT systems (obtained from literature), and applying the framework to the development of a centre pivot irrigation system (CPIS). This means that the research has two outputs: 1) the usability framework, and 2) the CPIS case study. It was seen that many of the usability heuristics call for the proper implementation of systems engineering. A systems engineering approach with effective contextualisation, constructivism, complexity reduction and communication establishes a usability baseline for IoT systems.

1.1 Document overview

The research methodology is discussed in Chapter 2, after which the problem statement is discussed in Chapter 3. The literature study follows in Chapter 4, where the research challenges are identified and analysed, and the research solutions are also identified. Chapter 5 is the development and validation of the usability framework with implementation on the

development of a CPIS. The study is concluded in Chapter 6 and recommendations given. Lastly, Appendix A is the peer-reviewed IEEE article, including the acceptance email, as the article is scheduled to be published later than the submission date of this study.

2 RESEARCH METHODOLOGY

The research methodology used in this research is aimed at delivering a design artefact as part of its deliverables. This research is considered to be applied engineering research as the focus is on providing practical value to a design environment.

Research is an inductive and deductive process where observations and theories are used in order to form a problem statement, possible reasons for the existence of the problem, possible solutions, and finally the validation of the solution. Figure 1 shows how observations and theories are related through induction and deduction. Different research methodologies use different entry points and outputs in the research cycle, each with their own merits. Three research methodologies are discussed in this chapter as the combination of the three methodologies provides a way to deliver the required design artefact.

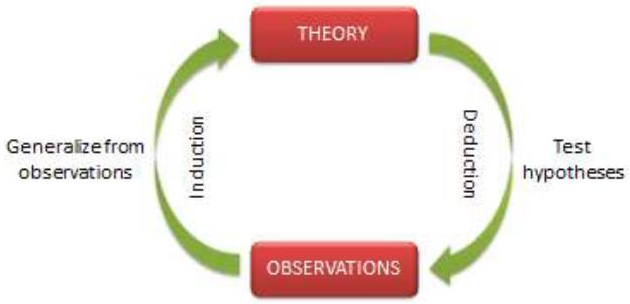

Figure 1: The cycle of research [1]

2.1 Design Science Research

The goal of Design Science Research (DSR) is to solve a real-world problem by generating an artefact and creating knowledge in the process. DSR is thus problem-based [2], [3], whereas other methodologies may be theory-based.

Figure 2 illustrates the DSR research paradigm. Information systems (IS) obtain input from the environment in the form of business needs and requirements. This includes people, organisation and technology needs [3]. IS also obtains input from a knowledge base. This includes foundational knowledge like theories and models and methodologies like data analysis and experimentation. With these inputs, artefacts are created and new knowledge added to the knowledge base. Throughout, new artefacts and knowledge are evaluated for validity. From this, the knowledge base and artefacts are refined and improved [2], [3].

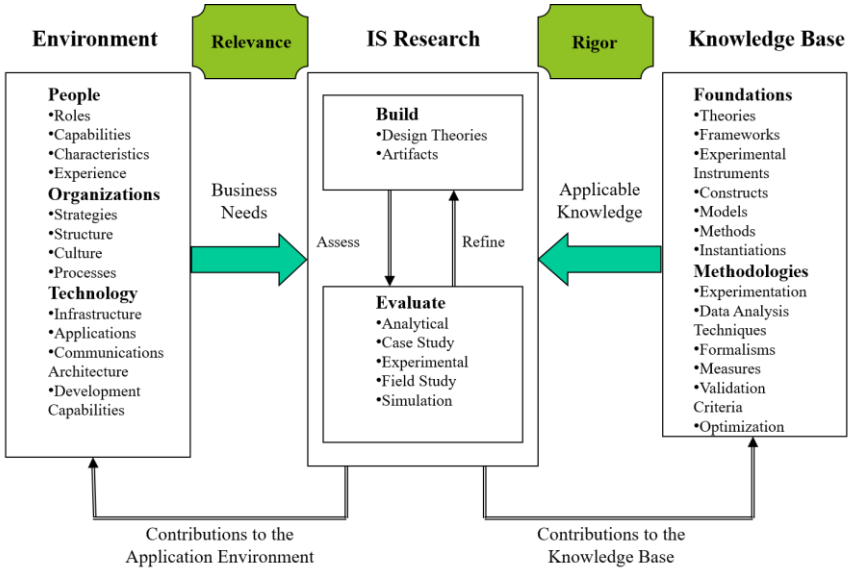

Figure 2: DSR research framework [3]

Thus, from a newly designed information system in the real world, knowledge is added to the knowledge base of the scientific world. The new knowledge and artefacts contribute to the environment [2]. This way, the environment, information system and knowledge base are connected in three cycles. The three cycles are shown in Figure 3.

The *relevance* cycle forms an interface between a newly designed system and the environment. It includes elicitation of product requirements at the beginning of a project and field testing at the end of that project. The *design* cycle is where designs are done from requirements, abstracted models are provided for theoretical analyses, and the final design is evaluated against test requirements. This cycle includes creating and evaluating both new knowledge and artefacts. The third cycle is the *rigour* cycle, which includes the application of grounded theories developed from theoretical analyses and obtained from literature studies, generation of new knowledge and provision of new knowledge to the existing knowledge base [2].

Guidelines for DSR are [3]:

- Design must be seen as producing an artefact;
- Problem relevance must be demonstrated;
- Design evaluation must be conducted;
- Research contributions must include improvement, at least;
- Research rigour is used to ensure a grounded design results;
- Design is often done as a search process;
- Communication of research must be done.

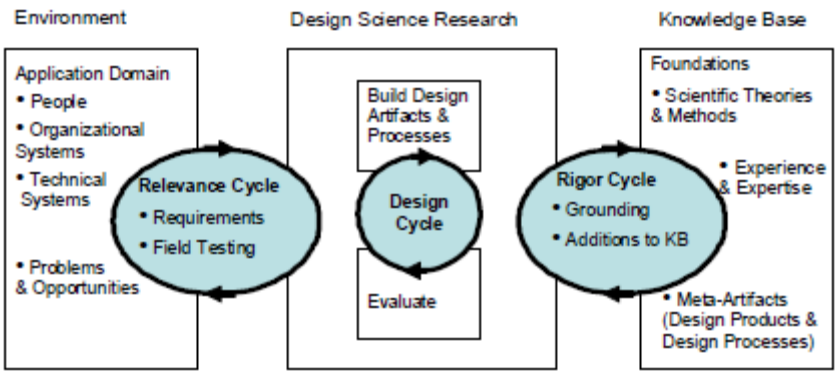

Figure 3: Design science research cycles [2]

2.2 Action Design Research

The goal of Action Design Research (ADR) is to solve a problem by generating knowledge whilst creating an artefact. ADR is thus knowledge-focused, whereas DSR is problem-focused [4]. ADR addresses two challenges, namely it:

- Evaluates and intervenes in problem situations that may occur in specific organisational settings, and
- Addresses a class of problems in a situation by constructing and evaluating an artefact.

The principles for ADR are as follows, namely that it [4]:

- Is practice inspired;
- Is theory ingrained;
- Includes reciprocal shaping;
- Has mutually beneficial roles,
- Depends on authentic and concurrent evaluation,
- Is based on guided emergence,
- Provides generalised outcomes, and
- Allows for artificial abstraction.

Figure 4 shows the build, intervention and evaluation (BIE) model of the ADR method. It starts by formulating the problem with practice-based research and a theory-ingrained artefact. The building, intervention and evaluation phase follows. From this, the reflection and learning phase follows. The knowledge from phase three is used in phases one and two. From this cycle, phase four follows where learning is formalised, and lessons are learned [4].

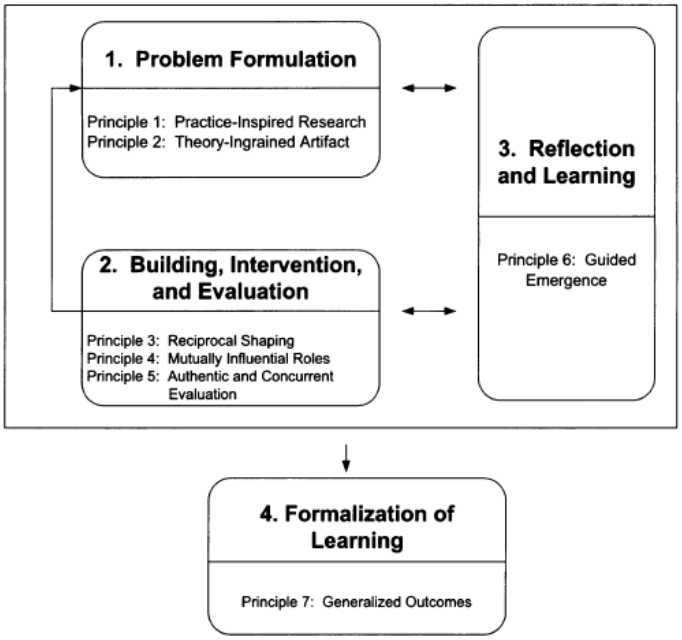

Figure 4: BIE model of ADR method [4]

2.3 Elaborated action design research

Elaborated ADR (eADR) is a combination of DSR and ADR that combines the advantages of each research method into a more efficient design science research methodology.

Design science research methodology (DSRM) is problem-based and uses a knowledge base to solve a problem (Section 2.1). Figure 5 shows the process model used for the DSRM part of eADR.

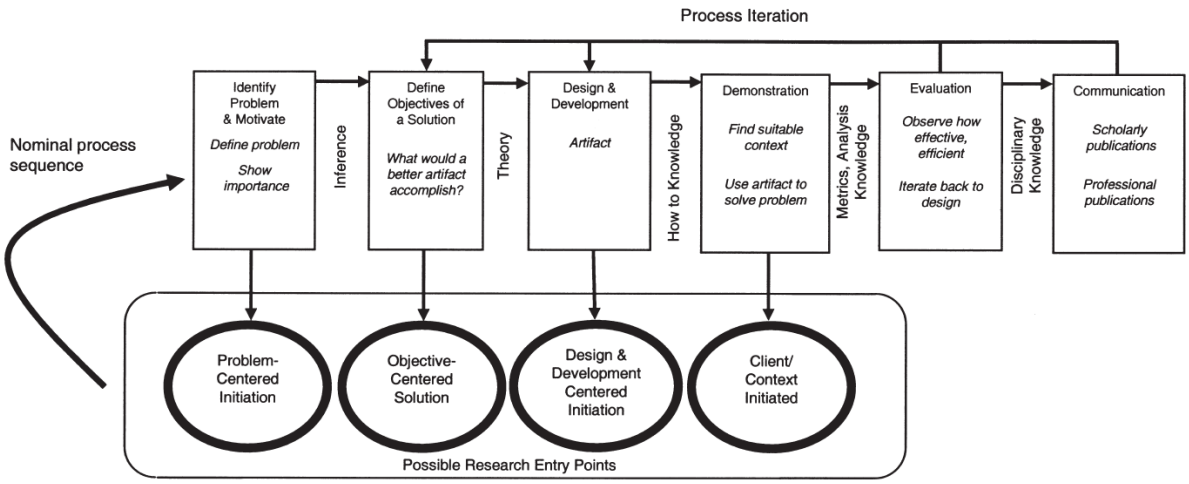

Figure 5: Design science research process model [5]

The steps for the DSR process in the eADR case are [5]:

1. Problem identification and motivation, where the problem is analysed, and proof of it being an actual problem is given. When the problem is better understood, the problem can be handled and solved more effectively;
2. Solution objectives definition, where the objectives and properties of a possible solution are defined. If the objectives are adequately defined from early on in the process, the solution will probably be more adequate later on in the process;
3. Design and development, where the product is designed (from a systems engineering perspective [6], a concept, preliminary and detail design). The design needs to be according to the solution objectives defined previously. The product is built after the design has been finalised. This includes a prototype and manufacturing of an artefact on a commercial scale;
4. The demonstration, where the artefact is used in a suitable context to solve the problem defined at the beginning of the process. The artefact should be tested to determine if the artefact offers an adequate solution for the problem;
5. Evaluation, where the efficiency and effectiveness of the artefact are tested. This determines if the artefact solves the problem stated at the beginning of the process. If not, the solution will have to be adjusted. Remarks on the evaluation are referred back to the design phase, and adjustments are made;
6. Communication, where scholarly and professional publications are made to add newly generated knowledge to the knowledge base. The communication should be as comprehensive and objective as possible to help prevent issues in future projects that will be based on the improved knowledge base.

From the evaluation and communication steps, feedback exists to the *solution objectives definition* and *design and development* steps for corrections and improvements. From the first four steps, there are outputs to research entry points. From each of these points, the current ADR stage is started. They are [5]:

1. Problem centred-initiation;
2. Objective-centred solution;
3. Design and development centred initiation; and
4. Client/context initiated.

To further the discussion on eADR, it is necessary to return to ADR principles and its application in eADR. As was discussed before, ADR is knowledge-based, where knowledge is, in turn, generated from developing an artefact, and the resulting new knowledge is added

to the knowledge base (Section 2.2). Figure 6 shows the ADR stages and the process model. The four stages are [7]:

1. Diagnosis, where the problem is analysed and the solution objectives defined;
2. Design, where the artefact is designed (conceptual, preliminary, and detail design);
3. Implementation, where the artefact is implemented in the problem context;
4. Evolution, where the evaluation is done, and suggestions are made for adjustments.

Each of the four stages above has five steps [7]:

1. P – Problem formulation/planning;
2. A – Artefact creation;
3. E – Evaluation;
4. R – Reflection; and
5. L - Learning.

In every stage, the five steps above are followed in a cyclic manner. Each stage has a research entry point. These entry points are the entry points as defined in the DSR process model above. From the current DSR stage, the entry point is used to begin the ADR stage accompanying that DSR stage.

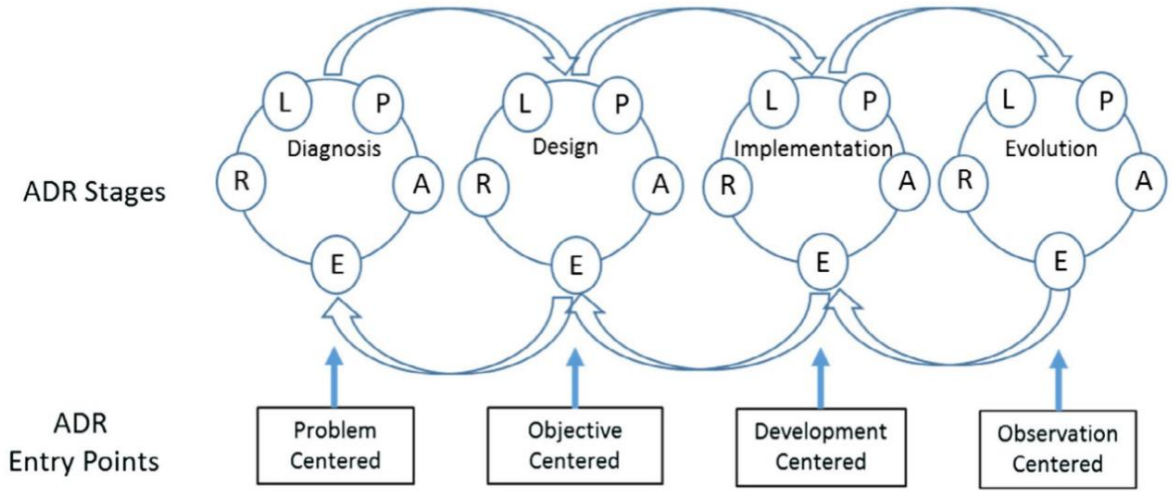

Figure 6: Action design research stages and process model [7]

Figure 7 shows an expanded view of the ADR intervention cycle and gives more clarity about the ADR process model, where the activities are also shown. Each of these cycles occurs in conjunction with the DSR process model. This means each cycle has a build and evaluation design cycle, including the artefact [7].

The process that is followed leads to the creation of the following knowledge, namely [7]:

1. Constructs;
2. Models;
3. Methods; and
4. Instantiations.

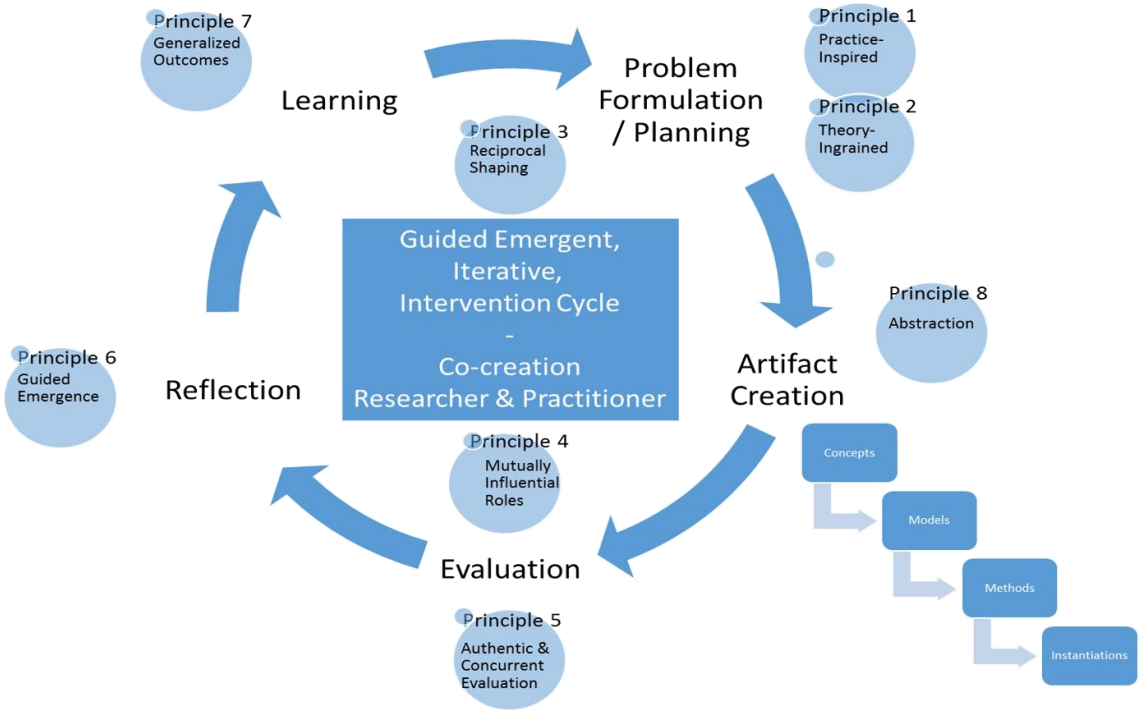

Figure 7: Elaborated action design research cycle [7]

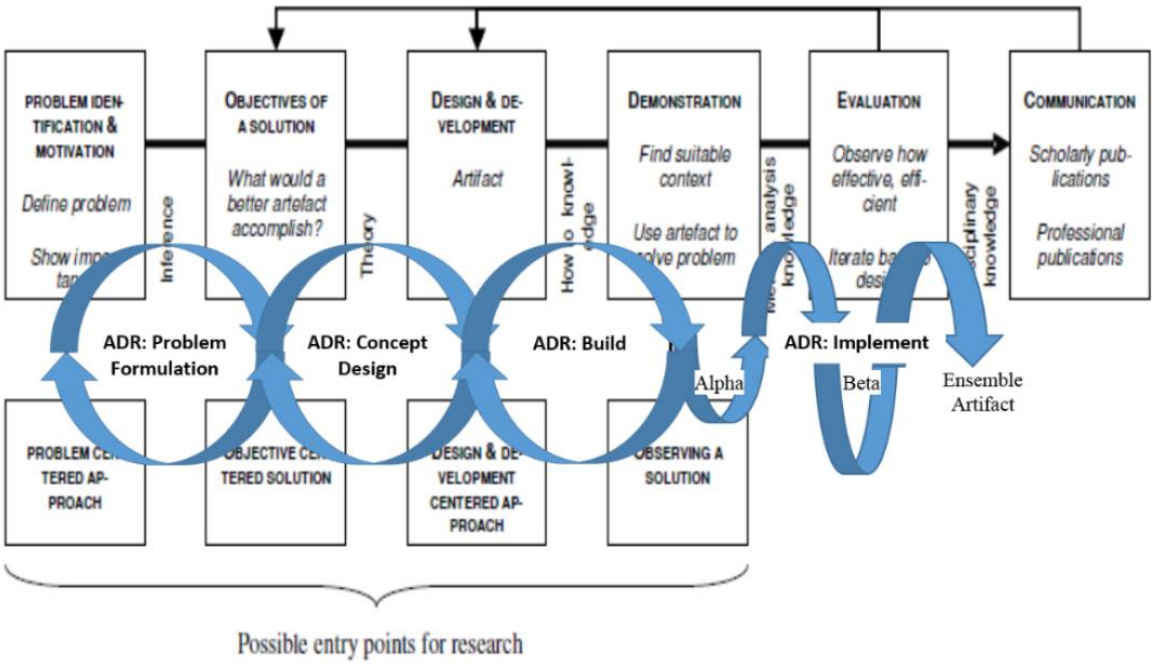

Figure 8: DSR and ADR combination for eADR [8]

The eADR cycle (as seen in Figure 7), is repeated for each of the DSR stages to produce an ensemble artefact. The combination of the DSR and ADR processes is shown in Figure 8. This shows the embedment of the ADR process in the overall DSR process. The two processes thus work together to generate knowledge and artefacts. This optimises the research process for better knowledge and artefacts as eADR is both problem and knowledge-based. Through this combination, every DSR stage is optimised in the way an artefact is created, the knowledge and artefact are evaluated and reflected upon, and how lessons are learned. Valuable knowledge is created and documented throughout the whole process, which will, in turn, assist future projects.

2.4 Conclusion

In this study, as it includes the development of an artefact as well as knowledge creation, eADR (both problem and knowledge-based) is the most effective research methodology to apply. Also, as the aim of the study is the creation of two artefacts, the DSR paradigm will be employed in conjunction with eADR. The artefacts that will be created include 1) a method to apply usability to the full life cycle in general, and 2) a centre pivot irrigation control panel, with usability knowledge and methods applied. The literature study acts as the knowledge base for the usability framework artefact. The case study acts as the practical environment where the usability framework is applied.

The knowledge elements generated in this study are 1) knowledge about the application of usability to the full life cycle of a product, and 2) knowledge about designing a centre pivot control panel, specifically for usability in an agricultural environment.

3 PROBLEM STATEMENT

3.1 Research Question

What will a full system life cycle usability framework for IoT systems comprise of as seen from the view of a case study into the usability of a pivot irrigation system?

3.2 Project Scope

3.2.1 Problem validation

From the research question, it can be seen that the keywords for this study are usability, full system life cycle, IoT systems, and pivot irrigation system. During the literature study (Chapter 4), various information sources were used to research the keywords and fields of knowledge related to them. Table 1 shows the information sources used in the literature study. From the literature study, five research challenges were defined. The five research challenges are also shown in Table 1, in the last row. An arrow pointing downwards (↓) indicates that the information source is used to validate the relevant research challenge.

Table 1: Problem validation

Information sources					
Published case studies			↓		↓
Work sessions				↓	↓
Published articles	↓	↓	↓		↓
Observations				↓	↓
Books	↓	↓			↓
Research challenges	1) Lack of information on usability in IoT	2) No clear definition of IoT in general	3) Usability over the full life cycle in IoT systems not defined	4) Usability in pivot irrigation controllers not fully applied	5) Focusses mostly on end-user usability

From the research challenges, it can be seen that usability in full IoT systems life cycles is lacking, including for pivot irrigation systems. Where usability is included, the focus is mainly on the end-users, instead of all system stakeholders and users. Thus, the five research

challenges validate the need for a full system life cycle usability framework for IoT systems, including pivot irrigation systems.

3.2.2 Research solutions

The following methods and objectives are defined for this study to address the five research challenges:

1. A grounded theory framework will be created that can be used as a general usability framework for IoT systems.
2. A general and clear definition of IoT is developed;
3. A usability analysis of the full general system life cycle is done and general usability issues identified. Generalised usability heuristics for IoT systems are developed from existing usability heuristics in literature. The usability issues are analysed for the generalised and product usability heuristics that can be applied to the issues as solutions;
4. A usability analysis of a centre pivot irrigation system will be done over the complete system life cycle. Shortcomings and improvements will be identified. The usability heuristics in point three are implemented in the design of the system and product;
5. A general and clear definition of usability is developed where the focus is shifted from the end-user to all users and stakeholders.

4 Literature study

The different sections discussed in the literature study include usability, systems engineering, Industry 4.0, IoT, system life cycle, ergonomics, centre pivot irrigation systems, and synthesis from literature. The relevance of each section is provided throughout the sections.

4.1 Usability

The user experience is important to consider when designing a system and its user interfaces. Usability is seen as a part of the overall user experience and affects all users and relevant technology. A usable user interface is more than just “easy to learn” [9] and includes many different aspects that are considered in this section.

ISO 9241-11 is an international standard for user interface usability design. It defines usability as: “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [10], [11]. There are other definitions of usability, each identifying important attributes. Some of the most popular definitions are from Nielsen [12]–[14], Shackel [15], Shneiderman [16], [17], Norman [17], ISO 9241-11 [10], and ISO 9126 [18]. These definitions are compared below to determine the most appropriate and all-rounded usability definition for further use in a generalised, system-wide usability framework.

Nielsen, Shneiderman, and Norman define usability design heuristics that will be analysed and discussed below. Furthermore, Nielsen, Shackel, ISO 9241-11, and ISO 9126-1 each define specific attributes for usability, as shown in Table 2. The attributes have many overlapping attributes like learnability, efficiency, effectiveness and satisfaction. Most of these attributes are intended to define the end user's experience of an artefact. But attributes like errors, efficiency, effectiveness, and operability can be extrapolated to the rest of the system users and the system full life cycle.

Table 2: Attributes for usability as per Nielsen, Shackel, ISO 9241-11, and ISO 9126

Nielsen [12]–[14]	**Shackel** [15]	**ISO 9241-11** [10]	**ISO 9126** [18]
Learnability	Learnability	Effectiveness	Learnability
Efficiency	Effectiveness	Efficiency	Operability
Memorability	Flexibility	Satisfaction	Understandability
Errors	Attitude		Attractiveness
Satisfaction			

Usability effectiveness is an attribute, or rather a technical performance measure that by definition, encapsulate all the other attributes considered to be design-dependent parameters [6]. Design dependent parameters, in this study, are thus the measurable attributes used to define usability effectiveness. Satisfaction, for example, is a qualitative human factor that can't be measured for design purposes and is not necessarily useful to include. Methods such as quality function deployment are often used to quantify quality factors by conducting interviews and tests [6] – these will not be considered within the scope of this research, but are noted for the sake of completeness. A qualitative view is taken on usability as the specific focus is on providing a system-wide, generalised definition of usability.

According to Wegge and Zimmermann [19], accessibility, usability and safety can be seen as components of ergonomics, which points to a focus on the end-user. Another way to view usability is that accessibility, safety and ergonomics can be seen as components of usability. This view gives a broader definition of usability that moves usability from end-user focus to a focus that includes all possible users. Ergonomics is discussed in Section 4.5.

Existing usability frameworks, including Folmer and Bosch [15] and Van Welie [16], represent layered usability frameworks, as seen in Figure 9 and Figure 10. In both frameworks, the usability properties are divided into four layers. These layers represent the 1) usability attributes, 2) usability heuristics, 3) activities, and 4) different knowledge types. Van Welie's framework is more simplified and uses the ISO 9241-11 definition, whereas Folmer and Bosch use at least four different definitions and more factors.

A possible issue with layered usability is that important information may get lost between the layers. According to Winter [20], another issue may be that the exact impact to an element may be unknown in the high-levels if a general framework or design is done.

Other usability frameworks and models consist of heuristics, guidelines, guessing analysis questions, or principles. Some examples of such frameworks were developed by Nielsen [12]–[14], Malcomb and Tharp, Polson and Lewis, Carroll and Rosson, Macintosh, Sunsoft [14], Shneiderman [16], [17] and Norman [17].

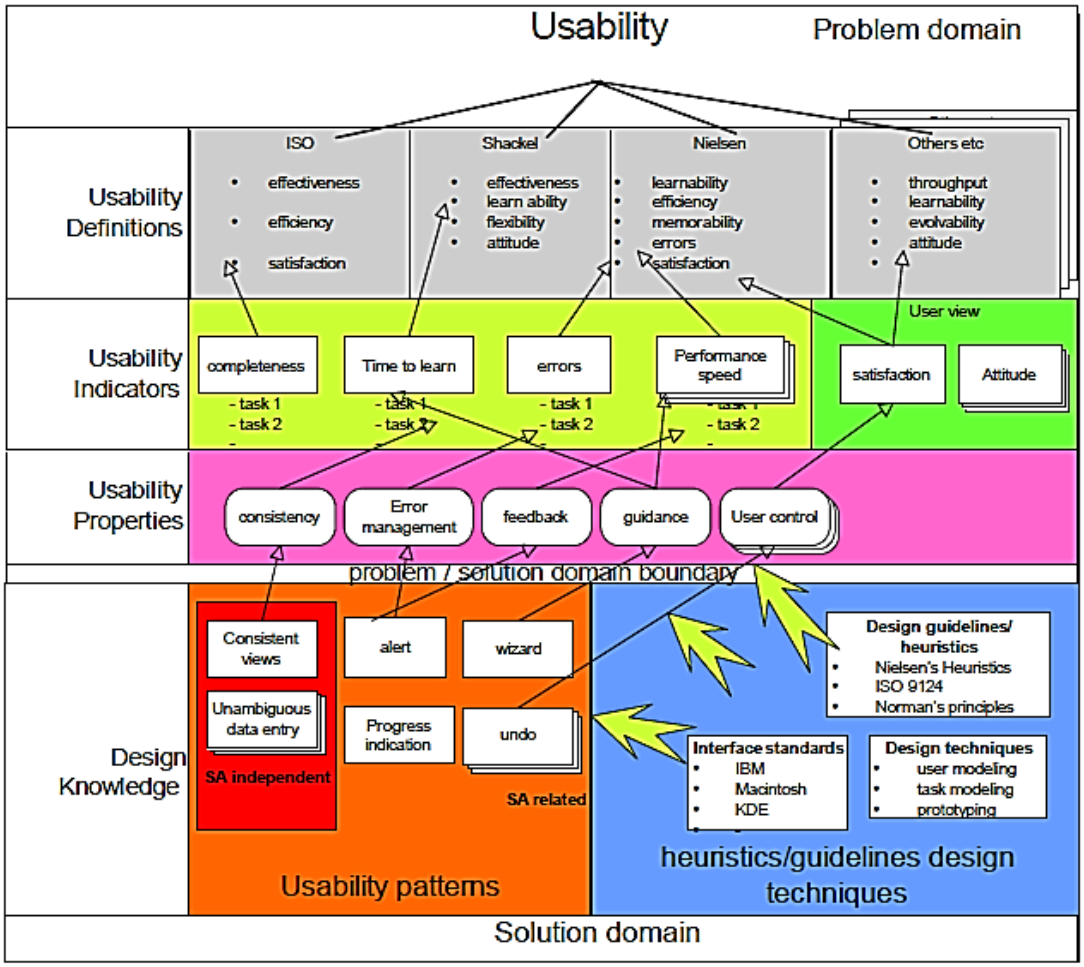

Figure 9: Folmer and Bosch usability framework [15]

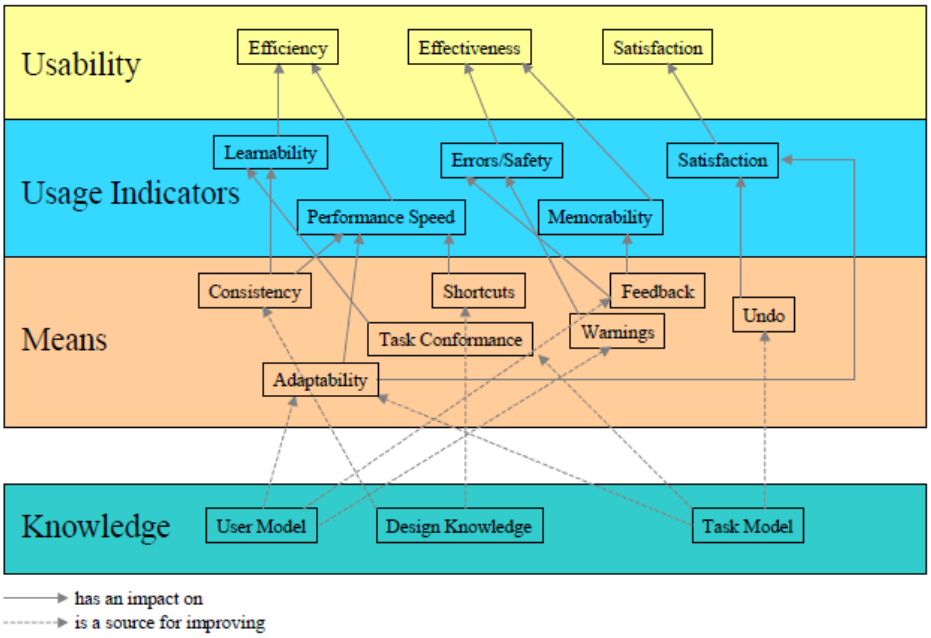

Figure 10: Van Welie layered model of usability [16]

The usability models above include requirements, design, and implementation phases, but typically do not include maintenance other than error management. Although usability models are applied in the internet and computerised systems, the connection to IoT is not clearly defined in the usability literature. But, since users and technology are required to interface across more complex interfaces, it is necessary to take a critical look at usability in IoT across a full life cycle. The models discussed above, especially concerning usability attributes (also, knowledge obtained from the case study considered in this research), do provide a baseline on how usability can be generalised to apply to systems over a full system life cycle.

4.1.1 Usability benefits

There are many benefits to a user interface that is usable and user-friendly. When users find a user interface learnable, efficient, memorable, error-tolerant, satisfying and effective, they will be more willing to use the user interface. In e-commerce, this will lead to more sales [21]. If a user finds a user interface satisfying, they may give it good ratings and recommend it to other people. This will increase application/device sales.

Having a usable interface have more benefits than user satisfaction. It reduces development and support costs and also reduces development time [21].

4.1.2 Usability heuristics, principles, guidelines and rules

Multiple heuristics, principles, guidelines, and rules are considered when evaluating the usability of a user interface. These heuristics are mostly applied to user interfaces, especially for mobile and internet applications.

Heuristic evaluation is an evaluation technique where the evaluator looks at the user interface and gives an opinion about the user interface. Heuristic evaluation is a common evaluation method used because of its time effectiveness and because the evaluator doesn't need expert knowledge [13].

Various heuristic, guidelines, principles and rules sets, have been developed by researchers, where these sets are usually used in the heuristic evaluation of user interfaces for end-users. These heuristics, principles, guidelines and rules may be extrapolated to all system users and a full life cycle, as opposed to being applied to only human-machine interfaces. Authors responsible for such sets are Nielsen [12]–[14], Malcomb and Tharp, Polson and Lewis, Carroll and Rosson, Macintosh, Sunsoft [14], Shneiderman [16], [17], Norman [17], ISO 9241, and ISO 9126. Nielsen's ten heuristics, Shneiderman's eight golden rules of interface design, and Norman's seven principles are most popular [22].

Some of the main heuristics, principles, guidelines and rules are:

- **Learnability**: Learnability is the time it takes a user to accomplish a task for the first time. In other words, it is the amount of time it takes a user to learn to accomplish a task [21], [23];
- **Efficiency**: Efficiency is the time it takes a user to accomplish a task while completing the action with accuracy [21], [23];
- **Memorability**: After learning to accomplish the task, the user has to be able to accomplish the same task after redoing the task after a certain amount of time [21];
- **Error tolerance**: Error tolerance is the number of errors, the seriousness of the errors and the efficiency with which the user can deal with the errors [21], [23];
- **Satisfaction**: Satisfaction is the emotional reaction of the user to the user interface. In other words, they like using the user interface or experience it as bad? Is the user interface engaging? [21], [23];
- **Effectiveness**: Effectiveness is the accuracy of which a user accomplishes a task. Not the time, but rather the numbers of errors a user makes [24];
- **Perceivability**: This includes text alternatives, time-based media, adaptability and distinguishability [24];
- **Operability**: This includes keyboard accessibility, seizure awareness and navigable [24];
- **Understandable**: This includes readability, predictability and input assistance [24];
- **Robustness**: This includes compatibility [24].

A brief overview of the heuristic sets is provided below.

Shneiderman's eight golden rules for interface design are [16], [17]:

- Strive for consistency;
- Enable frequent users to use shortcuts;
- Offer informative feedback;
- Design dialogues to yield closure;
- Offer error prevention and simple error handling;
- Permit easy reversal of actions;
- Support internal locus of control; and
- Reduce short-term memory load.

Norman’s seven principles are [17]:

- Use both pieces of knowledge in the world and knowledge in the head;
- Simplify the structure of tasks;
- Make things visible: bridge the gulfs of execution and evaluation;
- Get the mapping right;
- Exploit the power of constraints, both natural and artificial;
- Design for error;
- When all else fails, standardise.

Nielsen’s ten usability heuristics include the following:

- **Visibility of system status:** The user should always be informed about the current status of the system. This is done with specific feedback to the user [12]–[14];
- **Match between system and the real world:** Users don't understand codes, they understand words and phrases that are familiar to them. The system should use words and phrases, rather than system-oriented terms [12]–[14];
- **User freedom and control:** The application should give the user control and freedom in his/her actions. When they press the wrong button, there should be an emergency exit or stop [12]–[14];
- **Consistency and standards:** The system should follow conventions and be consistent. Consistency makes it easier for a user to navigate and use the application/system (less to learn) [12]–[14];
- **Help users recognise diagnose, and recover from errors:** The system should make it easy to identify and solve an error by the user/developer [12]–[14];
- **Recognition rather than recall:** Memory used by the user should be minimised. A user shouldn't have to use only memory to use a part when navigated from another. The user interface should be intuitive and with every part be easy to understand and figure out [12]–[14];
- **Flexibility and efficiency of use:** The system should be easy to adjust, time-efficient and promote user accuracy [12]–[14];
- **Aesthetic and minimalist design:** The system should have a minimalistic design. This means only relevant information and good visibility of the information [12]–[14];
- **Error prevention:** The system should be designed to prevent errors from the developer as well as the user side [12]–[14];
- **Help and documentation:** Customer support and documentation should be accessible if needed by the user or a way to contact the developers to report errors [12]–[14].

Nielsen proposed the ten heuristics in 1990 [13] and in 1994 [14] evaluated the ten heuristics by comparing it to other usability frameworks. After the evaluation, he removed “help and documentation”, but for this study, “help and documentation” will be essential for the full life cycle.

Both Shneiderman and Normans’ rules and principles are encapsulated in Nielsen’s usability heuristics. Nielsen’s usability heuristics also form the most encompassing set of usability heuristics, rules, and principles mostly used for user interfaces. These ten heuristics will form the basis of a more generalised set of heuristics applied to the full life cycle and are thus very important in this research.

4.1.3 Usability engineering life cycle

The usability engineering life cycle is a set of tasks done in a specific order when developing software. As software is a part of IoT systems, analysing the usability engineering life cycle adds to the understanding of usability in IoT systems and the systems engineering life cycle.

The steps in the usability life cycle cover the entire system development phase. The steps make it easier for a developer to create an application that is usable and functional. The main focus of the life cycle is the usability of the system and the user experience [25].

Figure 11 shows the engineering life cycle, as described by Mayhew [25]. There are three main phases: requirements analysis, design/testing/development and installation.

As with systems engineering and the system life cycle, the usability life phase starts with the requirements analysis. The design, testing, and development phase also line up with the design and testing phase from the system life cycle, including the conceptual and detail design stages. The installation phase lines up with the system operations phase from the system life cycle.

The usability life cycle, however, terminates after implementation, whereas the system life cycle includes maintenance and retirement phases. For usability to be optimised, the usability life cycle should be extended to maintenance and system retirement. The lack of usability phases after installation adds to the confirmation that usability is not present through the whole systems engineering life cycle. The focus of the steps in the usability life cycle focuses on the end-user.

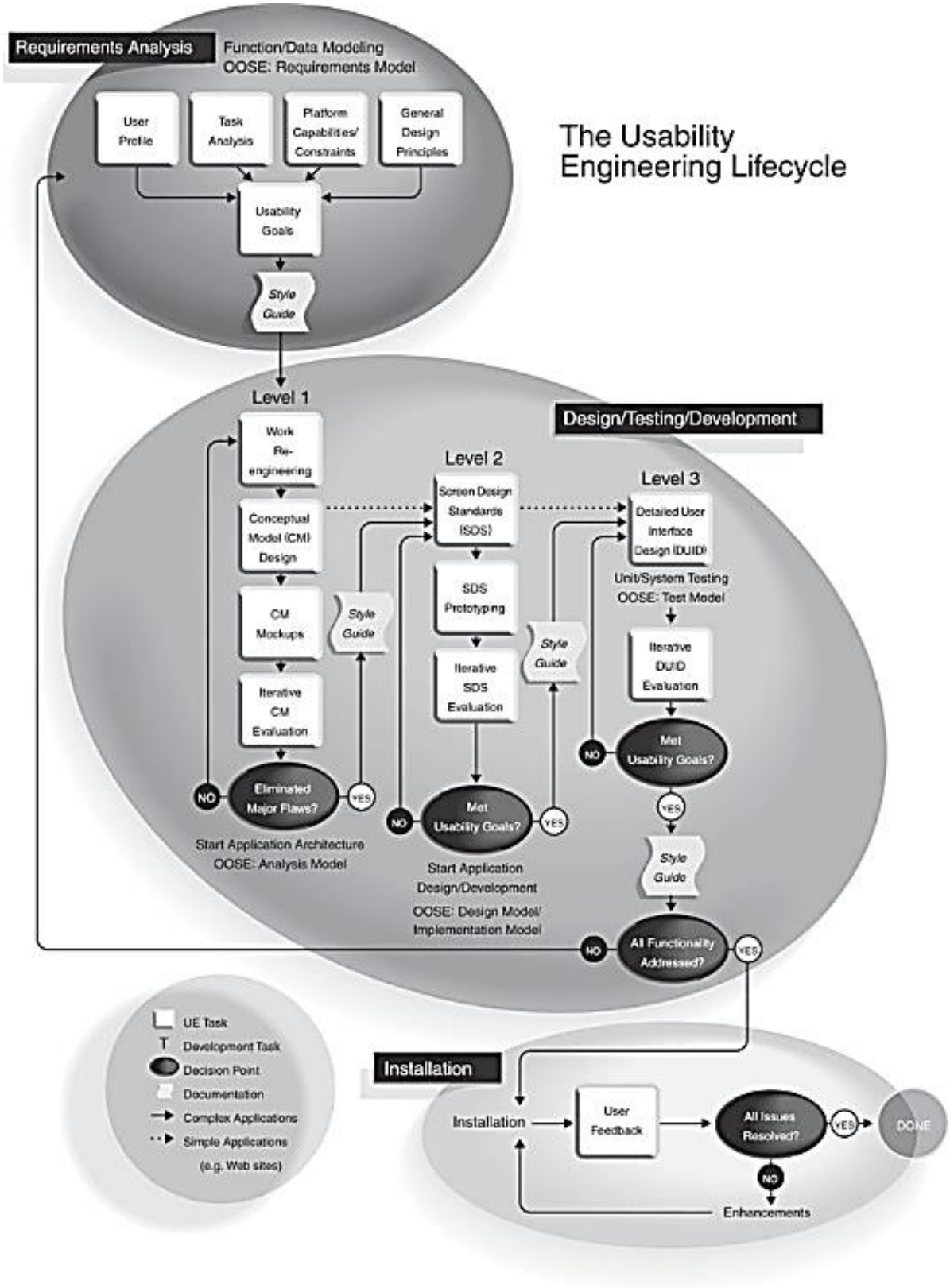

Figure 11: Usability engineering life cycle [25]

4.1.4 Usability issues and challenges in full system life cycle

Issues and challenges in the systems engineering phases were identified through literature. Identifying usability issues and challenges in each phase provide information that underlines the need for usability heuristics. The usability issues and challenges are used in Section 5.3 to show how usability heuristics can improve usability issues and challenges.

According to Blanchard and Fabrycky [6], the system life cycle can be divided into four main phases, as seen in Figure 12. For this study, the phases will be re-orientated and renamed to 1) requirements analysis, 2) design, implementation, and testing, 3) product manufacturing, 4) operations and maintenance, and 5) system retirement. The requirements analysis are classified as a phase since the usability characteristics are unique to this phase. All design stages (conceptual, preliminary, and detail design) are included in the design, implementation, and testing phase. System retirement is divided from the operations and maintenance phase. The stages in each phase are grouped according to similar usability characteristics, as well as similar usability issues and challenges.

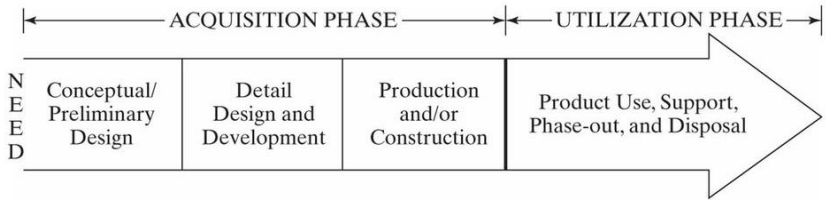

Figure 12: System life cycle phases [6]

4.1.4.1 Requirements Analysis

Issues in the Requirements Analysis might become worse through the phases if not sorted out early on. Requirements issues that are identified in later stages might be too costly to fix, and expensive adaptations are made [26]. The value of need analysis and requirements management (RM) is commonly underestimated and not given enough attention [27]. This section elaborates on these usability issues and challenges.

RM tools provide a platform for communication, traceability, change control, and information sharing [28]. The requirements engineering (RE) approach and tools should be agreed upon by all parties before starting the RE process. Otherwise, it may cause delays [29].

RM tools: The RM tools should be well-chosen. If the requirements for the tool are misaligned with the tool features, issues may occur [29]. RM tools often possess insufficient document and model-based coupling due to the variety of model-software [30]. Also, RM tools can't

necessarily incorporate large models and different documentation types [31]. When using an RM tool, a lack of information, training, and tool support [28] may cause unnecessary delays.

Information: Inadequate tracing, information access, tasks, goals, and documentation cause RM issues. Improper change management and version control are also problems [28], [31]. Distributing information is also a factor when team members are travelling. They need to take some information with them to work and import the new data and information afterwards [30].

Most requirements are given in text format, but some systems are too complex for only text-based information and require metadata models. The system diagrams give additional context and information [30]. Background information and a clear problem statement also give more context [30]–[32]. Dependencies between attributes, business objectives, etc. should be defined more clearly [30], [31].

Information security can also be a problem. Keeping your product information from the competition and giving suppliers information regarding product components without revealing too much can be difficult [30].

Human resources: Inadequate access to information regarding available human resources, including the competencies of the team members, may lead to inadequate planning and scheduling by management [28], [30], [33].

Transparency: Transparency regarding requirements, decision support, and system processes are insufficient. Distribution of information regarding made decisions by stakeholders also lack. This includes transparency about decision effects and high-level decisions [28], [30]–[34]. Progress and RE information are also not always frequently sent for verification to the client [29], [31].

Communication: A lack of well-structured documents, diagrams and relevant ontologies root misunderstandings between teams and different phases' stakeholders [30], [31], [35]. Language and cultural differences between the client and the development team can also be a factor. If the client and development team are not fully bilingual or one of them don't understand the mutual language sufficiently, a translator may be necessary [29].

The development team should have contact sessions with the client to acquire the necessary information regarding their sections of the development process [26], [29], [32], [36]. A lack of trust, strategy, and the distribution of requirements to different development teams/individuals give rise to poor communication, task clarity, context, and inadequate progress reports [28]–[30], [35].

Issues like inadequate follow-up and management issues may also occur when the roles and responsibilities of management and the client are not clearly defined and enforced [31].

Conflict: A lack of knowledge or information from either the client or engineering company result in conflicts. The development team should understand the needs of the client. The client, in turn, should understand the possibilities and constraints in technology, and that delays are common in the development process [29], [31], [35]. When RM issues occur, the responsible party does not necessarily take responsibility [29].

Flexibility: Flexibility is also necessary for the system and complete process [26], [36]. Poorly managed flexibility may result in impractical requirements [26].

Collaboration: Large development teams and organisations require more collaborative work and need more transparency and RM support [26], [28], [31]. Even if the current RM methods have issues, the current professionals may not always support and understand the necessary changes [31]. Changes to the current system might also be challenging to implement [33].

Risk management: Inadequate risk management leads to unidentified and unmanaged risks [34]. Risk identification and assessment are mainly done in the Requirements Analysis group phase and extend to the other group phases.

Requirements: The user requirements may be challenging to identify and define, especially for complex systems. This may include difficulties in determining or calculating the required time, costs, and resources [34]. Developers do not give non-functional requirements like maintainability and usability enough attention. Non-functional requirements are often challenging to design acceptance criteria for and assess. Existing test cases are also often academic and difficult to apply practically [30], [33].

Sign-off: Due to time restrictions, supervisors and managers sign off on information or documentation without reading and understanding it thoroughly. The information and documentation should be verified by a person who understands the information [29], [31].

4.1.4.2 Design, Implementation, and Testing

A whole design team is mostly part of the design process, rather than one individual designer. Every team member needs information from the requirements and other team members. Design, implementation and/or testing issues can be expensive to fix in later phases [26], [36]. This section elaborates on these usability issues and challenges.

Information: Insufficient requirements, inadequate work breakdown structure, development progress information, and task allocations affect task duration and information management. Change and configuration management are also common issues [27], [28], [33], [35]–[37]. A high frequency of changes in product design is difficult to manage [38]. A lack of version control causes additional work [28].

Inadequate input information, access to information, well-defined goals, responsibilities, methods, and strategies, affect team members' work [27], [28], [33], [36], [38], [39]. Incorrect assumptions and design errors lead to rework that could have been avoided if the task was well-defined from the start [27], [38]. Links and dependencies between objects or information should also be well-defined and maintained [30].

In model and simulation testing, acquiring test data for the tests can be difficult, especially for large data sets [40]. Information security is another issue and includes determining who has access to which information [37]. Distributing validated and correct information when team members need to travel is also lacking [30].

Resources: A lack of resources influence the planning capabilities for the project [27], [36], [39], [41]. Bad planning will cause issues in all phases, including inventory and personnel allocation. The inventory extends to the Product Manufacturing group phase [41].

Risk management: Inadequate technical risk management and improved implementation result in issues [34], [37].

Transparency and traceability: Inadequate transparency and traceability of information, requirements, task status, and documentation cause issues. This includes access to drawings, diagrams, information, and progress reports [27], [28], [33], [36], [39]. This also affects testing, as testing needs the information and requirements for verification and validation.

Standards: A lack of standardised documents, diagrams, drawings, models, etc., may cause difficulties in determining the status of the work done [27].

Validation and verification: The verification of information by specialists and stakeholders is lacking. The relevant information should be validated by the client [26], [27], [36], [37]. Validating units throughout the design prevents problems later on [42]. A lack of validating the product against the requirements causes discrepancies between the requirements and the final product [27], [38]. It happens that team members attend meetings unprepared and only present their work at meetings as a form of validation [27], [36], [38].

Decision support: Inadequate decision support and integrity are also issues that occur in systems. Decisions require information and permission from relevant stakeholders, including the client [27], [36], [37]. Additionally, choosing materials for the product can be difficult, but in some cases, the development team don't have better options [43].

Linking multiple decisions and their reasoning is a challenge in engineering projects [41]. A lack of follow-up on decisions leads to team members not implementing changes and errors being overlooked. Not all decisions made at meetings are written down, and some are consequently forgotten [38].

Communication: Different departments and teams often have different terminology and poorly written documents. Adding illustrations and diagrams will add context and avoid delays [28], [30], [35], [37], [39]. Regularly informing management teams and clients build understanding in the business world about the development process [27], [28], [33], [36]. Companies, cultures or development teams, or maybe even individuals, may try to hide issues and problems in the product design [37].

Different teams working in parallel and documenting information may cause redundancies or discrepancies in documents [30]. Different development methodologies may impose a problem when each team's section is incorporated and combined. This will be aided with frequent meetings [33], [34].

Collaboration: Lacking collaboration between development teams, specialists, management, manufacturers, clients, etc. occur. Inadequate information management, knowledge development, and integration cause issues and misunderstandings [26], [27], [36]–[38].

Large development teams and organisations require more collaborative work and have increased importance regarding DM support [28], [33]. A lack of collaboration between the development teams and the systems engineer will also cause problems on a system-wide scale. Complete models of the system enable the engineering company to analyse the entire system and identify problems early on [42].

The development team doesn't always possess some of the required knowledge and inadequate access to specialists. Integration and documentation of this specialist information also lack. Professional development for individuals will aid the flow of valuable information. Experience-based knowledge is also valuable and underrated in today's fast-moving technological world [26], [27], [34], [37].

Design management tools: Generating status and progress reports are features that are not generally available in current design management (DM) tools [28]. Additionally, related documents and models lack clear links in RM and DM tools. As with RM tools, incorporating coupling can be challenging [30]. Inadequate information distribution and development are also missing in DM tools [37].

Change implementation: Usability changes in the current development system are challenging to implement. Current company cultures delay usability changes by not agreeing to it [33]. Employees may distrust new methods, tools and procedures. Litigation also causes issues with implementing new procedures, tools and methods [37].

4.1.4.2 Product Manufacturing

The manufacturing team and other organisations included in the manufacturing process need a lot of information. Traceability, communication, and change management are essential in product management. Issues with product manufacturing cause costly manufacturing delays, as well as operation and maintenance issues. This section elaborates on these usability issues and challenges.

Planning: Waiting times, a lack of resources, the movement of materials, components, and product sections, and inadequate safety stocks cause delays in the manufacturing process [37], [41]. The main cause is a lack of planning, but labour strikes, transportation issues and other unplanned issues add the delays [41].

Communication: Poor communication, including different ontologies between design, manufacturing, operations, and maintenance departments, is a general problem in the manufacturing phase [35]. Decisions are made by developers and manufacturing firms and inadequately distributed to the relevant stakeholders [27]. Inadequate change management leads to issues during manufacturing processes and the procurement of incorrect components and parts [38].

Collaboration: Collaboration between manufacturing and design departments, and the client are not always sufficient and may lead to misunderstandings and manufacturing errors [26].

Manufacturing requirements: Separating design and construction leads to manufacturing issues. Manufacturing requirements are not taken into account during the requirements and design phases [27], [35]–[37]. This will require more comprehensive design information and models. In planning for manufacturing, a lack of integrating the work-processes and information will affect the following phases [37].

Progress in product construction can be challenging to monitor, control and evaluate [43]. A complex system of electronics and devices becomes difficult to manage efficiently [40]. There is a need for more robust, controllable, reliable, and transparent production systems. This includes the development and advancement of cyber-physical systems. These systems need to keep up with the dynamic environment, synchronise sub-systems, and create a symbiosis in human-machine-robot systems [39].

Information: Inadequate integration and synchronisation in work processes and information lead to unavailability of data and information, including the “as designed” and “as manufactured” models [37]. There will always be discrepancies between the two models, due to the standardisation of component values, tolerances, out of stock components, etc.. Inadequate information access, visualisation and transparency cause delays in the manufacturing process and errors in the manufactured product [27].

Environmental footprint: Reducing the environmental footprint of the manufacturing system is not prioritised [40], [44]. Some manufacturers think that effective energy management is not possible, especially in high energy consumption environments, like factories. This incorrect assumption leads to unnecessarily high energy bills and manufacturing costs [40].

Risk management: In the manufacturing phase, risks should be well-managed. Unidentified problems that arise should be managed sufficiently [34].

4.1.4.3 Operations and Maintenance

The product installation, day-to-day operations, and maintenance of the product should be done as effectively as possible. The training of the installation personnel, end-user, and service personnel are essential. This means communication and information sharing are usability factors that are key to the success of this group phase. This section elaborates on these usability issues and challenges.

Information: A lack of planning, information integration, and work-procedures definitions results in inadequate information models, information flow, and information management [37], [40], [44]. A lack of technical information regarding the product causes a lack of guidance for the end-user, service technicians, field personnel, and specialists [44].

Inadequate control, standardising, and linking of information affect the user’s access to knowledge. For example, in a disaster situation, access to fast, reliable, and real-time information updates are crucial [45]. Additionally, information security is becoming an important issue, especially attacks on private networks with sensitive or highly-sensitive

information. Vulnerabilities in the software are targeted by attackers and acts as a gateway into the system [40], [42].

Some data processing methods may be inaccurate [40]. Collecting data can be time-consuming and labour-intensive. The data collected can also be inadequate or of low quality [40].

Decision support: Inadequate decision support will lead to bad decisions. This is mostly due to incorrect or inadequate information, or maybe even inadequate access to information [45]. User data can also be difficult to access, manage, or understand, which complicates decision making, learning product operations, and product improvements [40].

Flexibility: Maintenance plans usually don't account for dynamic system environments. Maintenance management should be flexible enough for changes in the operational environment, required function, and operational conditions [40], [44]. Improvements should be easily implemented on the current product, which includes software updates [44].

Maintenance: Maintenance is misinterpreted as only repair work or preventing trouble. This image of maintenance affects developers and end-users to underestimate the value of efficient maintenance. Unfortunately, the numerous factors involved in maintenance complicate maintenance plans [44]. Regular maintenance schedules are not always kept by the users, as users miss or perform inefficient maintenance sessions [40].

Maintainability is lacking in system requirements and not adequately addressed in the system designs. Limited sensor and information processing possibilities also limit preventative and correctional maintenance capabilities [26], [44]. Additionally, designing for simplified assembly and disassembly is also insufficient [44].

Risk management: Risk management in the Operations and Maintenance group phase is essential for proper operational success [34]. Part of risk management is implementing proper maintenance.

A large number of sensors and devices makes the system more complex and difficult to manage risks [40]. Inadequate device configuration increases the risk for product operation issues. Automated configuration, if possible, will prevent an uninformed user from making configuration errors and simplify the installation process [45].

Diagnostics: Weak points in the design and/or manufacturing process are shown in a deterioration and failure analysis [44]. Insufficient reliability, quality control, resource

allocation, task-supervision, coordination, control, and scheduling are common issues in products. Some products can't cope with fast-changing environments and systems. Data management usually lacks in fast-changing environments [45], [46].

Determining the rate of time-based maintenance can be difficult, as deterioration isn't necessarily constant. It depends on many factors, including the operational environment. Additionally, incorrect measurements and diagnostics may lead to inadequate preventative maintenance [44].

When the system experiences a failure, the severity and quantitative evaluations of the failure may be challenging to determine. Failure analysis can be a time-consuming process and require specialist consultation. A lack of information and data causes difficulties in failure analysis [44].

Collaboration: The developing team(s), client, end-user, technical and installation services, field services, warehouses, monitoring specialists, and personnel need to improve collaboration for effective operations and maintenance [26].

Environmental impact: Products aren't generally designed to be material and energy-efficient. When the use of materials, resources, and energy input are minimalised, stakeholders, maintenance processes, and the environment will benefit. Less energy and consumed resources lead to lower operations and maintenance costs [40], [44]. There are multiple environmental impact assessment methods. Deciding which to incorporate can be a difficult task for the developing team [43].

4.1.4.4 System Retirement

Communication and personnel training is important with the dismantling and removal of the product. Products should be designed for environmentally friendly removal, recycling, and disposal. The absence of the original design information may create issues with re-engineering, as the new design team will have to "reverse-engineer" the new product from the information they can access. This section elaborates on these usability issues and challenges.

Designing for retirement isn't given enough attention. This includes safety and standards for retirement, disassembly, disposal, and recycling [43].

Disassembly: Disassembling a product can be challenging because of the complexity of the product itself and/or because the developers didn't sufficiently design for assembly and disassembly procedures [44].

Return value: When dismantling the product for materials and parts to dispose of, recycle, and/or reuse, the condition and quality of the materials will differ for each product unit. This results in fluctuations on the return value of the materials for disposal, recycling, and/or reusing. Difficulties in determining the available lifespan of a part is also a problem [40], [43].

Environmental footprint: Products require a smaller environmental footprint, even after retirement. The design should use materials and components that can be recycled and reused. There unfortunately also need to be a balance between the requirements, costs, and profits. This balance is hard to keep [40], [44]. It may also be challenging to convert environmental principles into design principles that can be integrated into the design [43].

Re-engineering: Product information and usage data are often lacking or hard to access for re-engineering purposes. This means that the re-engineering team needs to reverse-engineer the improved product from vague and unclear information, as well as from the product itself [44].

The identified issues in the five group phases point to usability not being appropriately utilised throughout the full system life cycle. Finding solutions for these issues will assist in the creation of the usability framework. The usability issues not only indicate an end-user focus but also indicate the possibility of expansion to all other stakeholders.

4.1.5 Usability evaluation

Evaluating usability determines whether the artefact is usable. If an artefact does not comply with usability standards, the design of the artefact should be re-evaluated. There are three main evaluation methods [15]:

- Testing;
- Inspection; and
- Inquiry.

When testing the artefact for usability, representative users or volunteers are asked to make a list of activities using the product or a prototype. The users' responses to the user interface are determined. This can be done with observation, co-discovery, learning, coaching, asking questions, retrospective testing, making the users think out loud while doing the activities [15], or measuring keystrokes.

The inspection of an artefact for usability includes usability specialists, users, and/or other specialists like software developers. They determine whether the usability of the artefact or prototype follows specified guidelines, heuristics, principles, or rules. This is done by

determining if the heuristics, rules or guidelines are incorporated correctly through inspecting the user-interfaces or using checklists to measure compliance [13], [15].

Inquiry requires usability evaluators that gather information about users' impressions of the artefact or prototype. This includes the users' likes, dislikes, preferences, and understandings. It can be done by means of field observations, interviews, surveys, logging use, or questionnaires. There are ethical principles that need to be incorporated into inquiries, especially interviews and questionnaires. There are many sources available for setting up questionnaires [15], like Questionnaire for User Interface Satisfaction (QUIS) [47], Perceived Usefulness and Ease of Use (PUEU) [48], After Scenario Questionnaire (ASQ) [49], etc.

Most of these evaluation methods focus on the end-user, but methods like heuristic evaluation can be extended to include all system users. This indicates that the usability of the heuristic evaluation will be an adequate testing method for this study. The focus will be extended to all stakeholders and the full system life cycle.

4.2 Systems engineering

Systems engineering has been gaining popularity as engineering systems have become more complex. A system comprises different types of elements that are combined to form a complex or unitary whole [6]. A system may consist of three different environments, namely physical, organisational and social. The presence of the different environments introduces many different elements and places them as interdependent components that form a functioning whole [50]. Systems engineering provides an essential foundation for describing engineering systems and provides a method for defining the system building blocks used in this study.

According to the International Council on Systems Engineering (INCOSE), the definition of systems engineering is [51]:

> “System Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem: Operations, Performance, Test, Manufacturing, Cost & Schedule, Training & Support, Disposal. Systems Engineering integrates all the disciplines and speciality groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.”

From the INCOSE definition, it can be seen that system engineers focus on the technology side, as well as the business side in as far as system economics is concerned. They bring all physical, organisational and social components together throughout the full system life cycle in an orderly and managed manner to ensure the successful design, production and operation of a product. Systems are becoming more complex, and systems engineering is needed to manage the complex system and its associated system risks [51]. Systems engineers need to be cognisant of engineering ontology, as well as the client (or business) ontology. The systems engineer essentially forms an interface between the client and the development team(s), and most importantly, also interfaces to a project manager. As part of the many systems engineering functions [10], requirements management and communication are critical when defining desirable usability characteristics.

Some existing models and frameworks focus on systems engineering, either in use or just theoretical.

- Davis [52] defined the socio-technical systems thinking architecture, where the focus is on predictive systems thinking.
- Carayon [53] defined advanced socio-technical thinking. The focus is on the end-user, the organisation and usability.
- Wilson [54] defined a version of the traditional system ergonomics and human-factor (E/HF) model. The focus is on technology and the end-user.
- Dul et al. [50] defined another E/HF strategy. The focus is on system definition.
- Dillon [55] defined some socio-technical concepts in IT systems, where the focus is on usability in socio-technical systems.

In terms of usability, only advanced socio-technical thinking, and socio-technical concepts in IT systems comply. None of the models and frameworks represents the whole system life cycle.

4.2.1 General Systems Thinking

The different environments and elements in a system are analysed for identifying interfaces and the flow of information in the system. This information is used throughout the study in the architecture and functional flows of systems.

As discussed above, systems consist of a complex combination of a physical environment, organisational environment, and social environment. The physical environment contains physical artefacts that can be demonstrated. The organisational environment defines activities

with their control and organisation (management structures). Finally, the social environment is characterised by people, their cultures, behaviour and habits [50].

A socio-technical system considers social as well as technical aspects of a system in general, where Figure 13 shows an illustration of such a system. The system has goals, stakeholders and users (people), buildings and/or infrastructure, technology, culture, and processes and/or procedures that all collaborate and influence each other. This internal system has external influences from an economic environment, regulatory frameworks and organisations, and external (indirect) stakeholders [52]. Engineers traditionally focused on the technical aspects of a project as systems are often complex and the system environment is prone to change, hence the need for a system boundary that sometimes excludes social elements of the environment. The need is for engineers need to think bigger than just the technical aspects for an efficient, effective and satisfying system. Systems engineers thus also play a significant role in designing a successful socio-technical system.

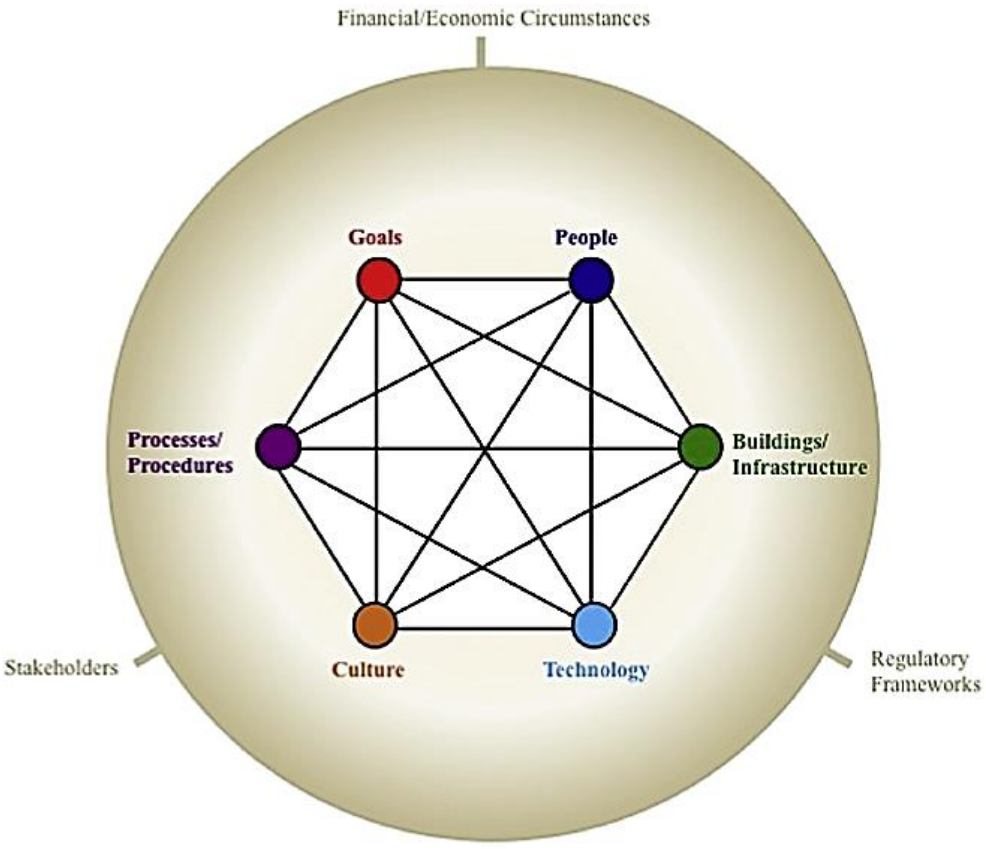

Figure 13: Socio-technical systems layout [52]

In a system, there are also different types of stakeholders and users. They can be divided into four major groups [50]:

- **System actors:** All persons that are directly or indirectly influenced by the design and performance. This includes employees and system users;
- **System experts:** Professionals and specialists that are part of the design team. This includes ergonomists;
- **System decision-makers:** Managers and clients with the authority to make decisions about the requirements, design, product purchasing, implementation and use;
- **System influencers:** People and organisations like the media, governments, standardisation organisations, regulators, trade and worker unions, and citizens with a public interest in the system. In short, it is people and organisations with influence from the outside.

Each of these groups can be divided into classes, including individual, company, country/region and world. *Individual* contains direct stakeholders and users. *Company* contains direct and indirect stakeholders and users in that company or organisation. *Country/region* is confined to stakeholders and users in the country, state, or region. *World* contains indirect stakeholders and users on a broad scale [50].

Systems engineering is used as a reference baseline for the development of the usability framework since systems engineering considers a full life cycle process and all associated resources. The engineering life cycle forms the timeline for the engineering process used in the usability framework, the case study, as well as the users' information.

4.2.2 System life cycle

In this study, usability is researched over the whole system life cycle. To this purpose, the system life cycle phases and steps should be investigated. The development of a system follows a full life cycle [6], as shown below:

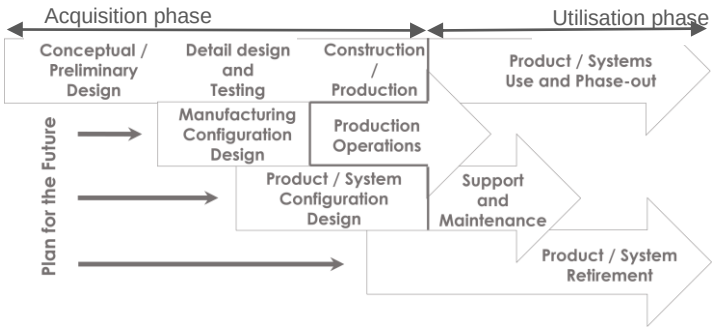

Figure 14: System engineering life cycle [6]

Figure 14 shows a high-level system engineering life cycle. The needs analysis phase is not shown in Figure 14, but it is located at the beginning of the system engineering life cycle. The needs analysis indicates if there is a need for the proposed artefact, and act as a foundation for requirements analysis. The needs definition includes the delivery of a project charter and feasibility study, amongst other important documents (such as a high-level user requirement statement, and others). This means that there is significant effort before a project is officially started [6].

The requirements analysis and specifications phase is also not shown in Figure 14 but is located between the needs analysis phase and the *conceptual* design. In the requirements analysis, all stakeholders' constraints are taken into account in the requirements and specifications phase as part of usability analysis. The *preliminary* design consists of a context diagram, states and modes diagrams, architectures, operational flows and resource allocations. This is the process of starting with the big picture and considering usability and external factors as well. The preliminary design also includes the mock-ups and considering different design options. From this, the final *detail* design is created. [6]. Usability heuristics should be factored into the requirements analysis and design process, but are generally considered mostly for human-machine interfaces.

In the production and/or construction phase, a product or system is produced, tested, and configured. Every unit must be functional, with its performance fully verified [6]. Usability should be factored into the manufacturing process as well as ergonomics such as anthropometrics and safety. Ergonomics and anthropometrics are generally included. Still, usability heuristics do not receive as much attention (i.e. often production facilities are designed to support floor personnel, but higher-level considerations do not always receive as much focus).

During the utilisation phase, a product is distributed and installed or constructed at set locations and operationally utilised as defined in the acquisition phase of the life cycle. This phase includes business aspects such as commercialisation and selling products [6]. Commercialisation includes logistics needed to get the product on the market, the sales price, warranties, and determining the channels of product distribution. Market specialists and business analysts are essential in the commercialisation phase [56]. Usability is mostly factored in for the end-user but should always be expanded to the distributors, installers, support, and maintenance personnel. All these factors should be included when the product is designed, but are not always.

When the decision for system retirement is made, it may include recycling and re-engineering. This is also important to prevent harm to the user and the environment [6]. In re-engineering, the necessary documents and information should be available to the re-engineering team, which is not always included in initial specifications and plans. If the need for the re-engineered system is unclear, the definition of need must be repeated at the end of a system life cycle, before an upgrade is done [57]. Usability factors such as access to original design information, decision support information, risk analyses and others are not always present, making the system less usable from a system perspective.

As the aim of the usability framework is to expand usability to cover the full system life cycle, usability should be highlighted in more phases of the system life cycle, including all users, not only end-users. The product should thus be designed for improved efficacy in all life cycle phases. The focus should also be extended to all stakeholders and users. By understanding the different life cycle phases, gaps in the general definition of usability will become visible and can be addressed. The focus of this study is, thus to take a general system-wide view on usability. A specific view of an agricultural IoT system is also taken.

4.3 Industry 4.0

Industry 4.0 or Industrie 4.0 as it is used in Germany, is the technical name given for the fourth industrial revolution. The characteristics of Industry 4.0 provide the foundation for IoT and are also featured in this study.

In some parts of the world, there are similar concepts named Industrial Internet, Integrated Industry, Smart Industry, Smart Manufacturing [58], Industrial Internet-of-Things (IIoT), and Internet-of-Everything (IoE) [59]. Industry 4.0 promotes increasing operational effectiveness. It also promotes the development of new business models, products, and services. Industry 4.0 also focusses on improving manufacturing and logistics [58]. Whereas IoT describes a general framework for a hugely interconnected system of elements, Industrie 4.0 was used to describe the German initiative to improve the manufacturing industry by using advanced manufacturing solutions (including information technology).

Advancements in the third industrial revolution started improving industrial processes like manufacturing, engineering, economic and environmental factors, and even the supply chain and system lifecycle. The fourth revolution was identified as recently as 2011, when the name Industry 4.0 started to circulate [58]. Throughout the four industrial revolutions, the role of humans has changed, often with a focus on technology that can replace error-prone humans.

Advancements in technology (like sensors and actuators), as well as advancements in data analysis, especially big data analysis and abilities, are larger factors that fuel the fourth industrial revolution [59].

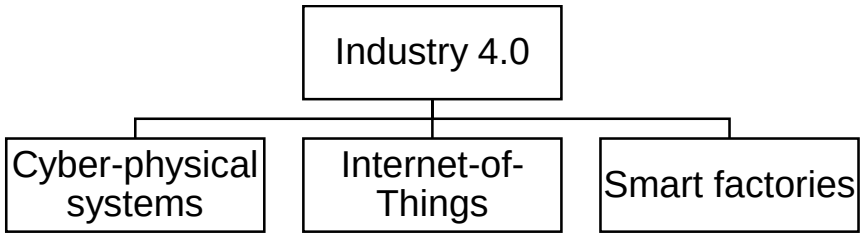

Figure 15: Industry 4.0 components [58], [60]

Figure 15 illustrates the main components of Industry 4.0, as defined by Hermann [58] and Alcácer [60]. The combination of these components is used to improve manufacturing, logistics and the system lifecycle [58].

Cyber-physical systems (CPS) is the combination of the physical and computational world, where computers and embedded technologies control the physical world and processes to improve the design and operational processes [61]. CPS provides the technology (“things”) for IoT. CPS has three main groups. Identification technology, like radio-frequency identification (RFID), provide the names and addresses to the elements in the system. Storage and analysis provide computational abilities with memory, databases, algorithms and/or artificial intelligence. Lastly, sensors and actuators provide the measurements and parts that are controlled [58].

IoT is an integral part of Industry 4.0 as “things” are connected to the internet. These “things” (CPS) include sensors, RFID, phones, and even appliances. These objects are connected to the internet using addressing schemas that can then connect and work together to achieve a specified goal. These objects work together using software and algorithms to control an artefact and/or achieve this goal, which may be improving the business and societal outcomes of an artefact or system.

Smart factories are where CPS and IoT are constructed, and industrial processes occur [58]. Smart factories are defined by Radziwon [62] as:

> “A Smart Factory is a manufacturing solution that provides such flexible and adaptive production processes that will solve problems arising on a production facility with dynamic and rapidly changing boundary conditions in a world of increasing complexity. This special solution could, on the one hand, be related to automation, understood as a combination of software, hardware and/or mechanics, which should lead to

optimization of manufacturing resulting in the reduction of unnecessary labour and waste of resource. On the other hand, it could be seen in a perspective of collaboration between different industrial and nonindustrial partners, where the smartness comes from forming a dynamic organization."

Besides factories, facilities also include smaller-scale manufacturing locations and may be extended to operational environments from a complete systems' approach.

According to Hermann [58], Industry 4.0 has four main design principles:

- Interconnection: Devices like sensors connect via the internet to interchange information and collaborate to accomplish a common goal;
- Technical assistance: Human jobs are changing from operators to monitoring, decision-making and problem-solving. Robots can perform unsafe activities including dangerous repairs, but still, need to be controlled and monitored by humans;
- Information transparency: High-context information is essential for making the correct decisions. The data analytics should be up to standard and available to all appropriate participants of the system;
- Decentralised decisions: Decentralised decision enable local and global systems to be done at the same time. For this, the interconnectedness of the system elements and the information transparency should be up to standard. Conflicts should be redirected to a higher level in the organisational architecture.

There are multiple models for industry 4.0, IoT and IIoT. Yang [63] identified three main domains, as seen in Figure 16. The technology level is comprised of technology like sensors, actuators, and user interfaces. The technology feeds raw data to the communication and networking level that "packages" the data using communication standards and protocols for the intelligence level. This is where storage and data analytics occur and produce information. The information flows through the communication and networking level to the technology level and controls the technology.

Some models expanded on the type of model Yang [63] developed to add business factors and more specific levels or domains. According to Gilchrist [59], IIoT's functional domains are business, operations, information, application, and control. The functional domains are illustrated in Figure 17. For the system to function properly, all the functional domains need to interact and communicate properly, as well as interacting with the physical system. The physical system and its interactions with the other domains are analysed in this study.

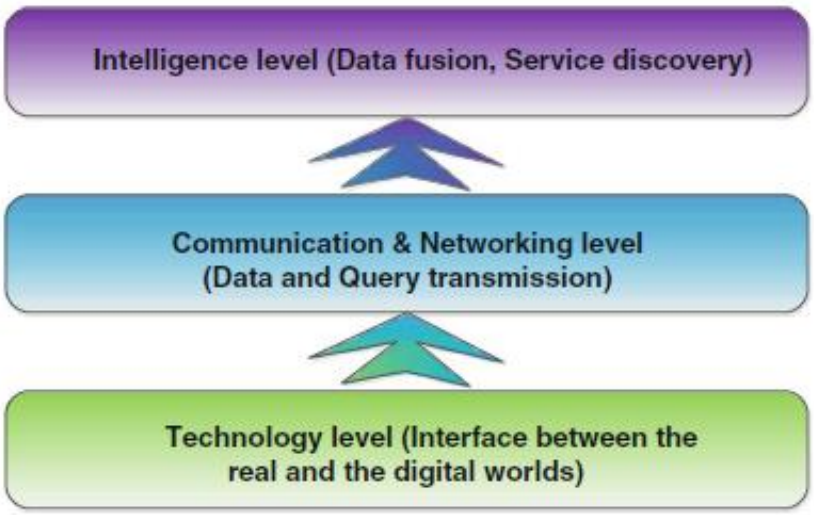

Figure 16: Three main domains for IoT by Yang [63]

The business domain contains business factors like enterprise resource planning (ERP), customer relationship management (CRM), warehouse stock management (WSM), etc. and incorporated these factors into IIot [59]. The other domains are engineering factors.

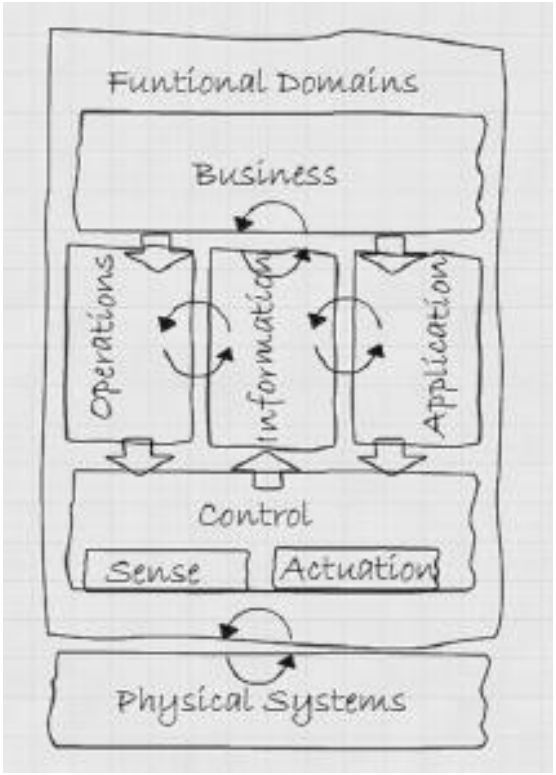

Figure 17: IIoT Functional domains [59]

Figure 18 shows the three-tier architecture of IIoT systems. These tiers show relationships with functional domains. Data and control are interchanged between the tiers to allow the system to function properly.

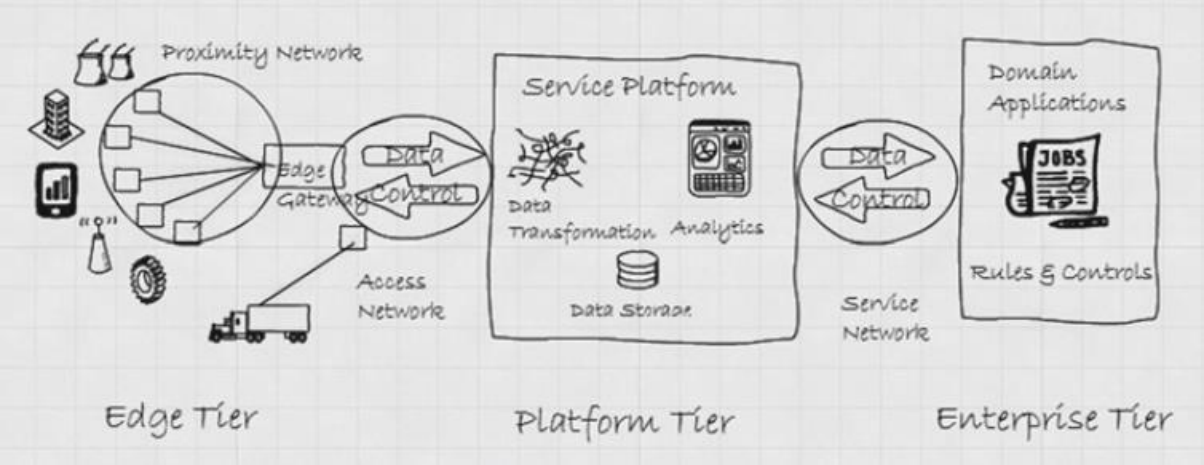

Figure 18: Three-tier topology for IIoT [59]

Another Industry 4.0 model used in practice is the Reference Model Industrie 4.0 (RAMI4.0). The RAMI4.0 model is shown in Figure 19. It is a 3D model with multiple layers on each axis. Most of the layers are similar to Gilchrist's model [60], [64], [65].

The business, functional and information layers agree with the functional domains in Figure 17. The communication layer follows communication standards and protocols to prepare data and information for the next layer. The integration layer provides the assets (sensors, RFID, user interface, etc.) the necessary information to digitise and control the assets [60].

Similarities to socio-technical systems are more evident in this hierarchy than in others, and the presence of a lifecycle is instructive. The layers reflect systems thinking where the physical, technical, and social domains are included, and an interdisciplinary approach is necessary for a healthy system. This can be seen in the layers where business, information, information, enterprise, work centres, and the product are all included. Systems thinking and Industry 4.0 characteristics, domains, and tiers are all included to form the layers in Figure 19.

While analysing Industry 4.0, it was seen that IoT is generally defined only in terms of physical artefacts and their communication, especially RFID and sensors. Besides this, Industry 4.0 still provides characteristics that are used in defining IoT in this study. Usability is not generally included in Industry 4.0 and IoT, as well as using a full life cycle perspective. Industry 4.0 does provide input into IoT systems and information flow that is used in this study.

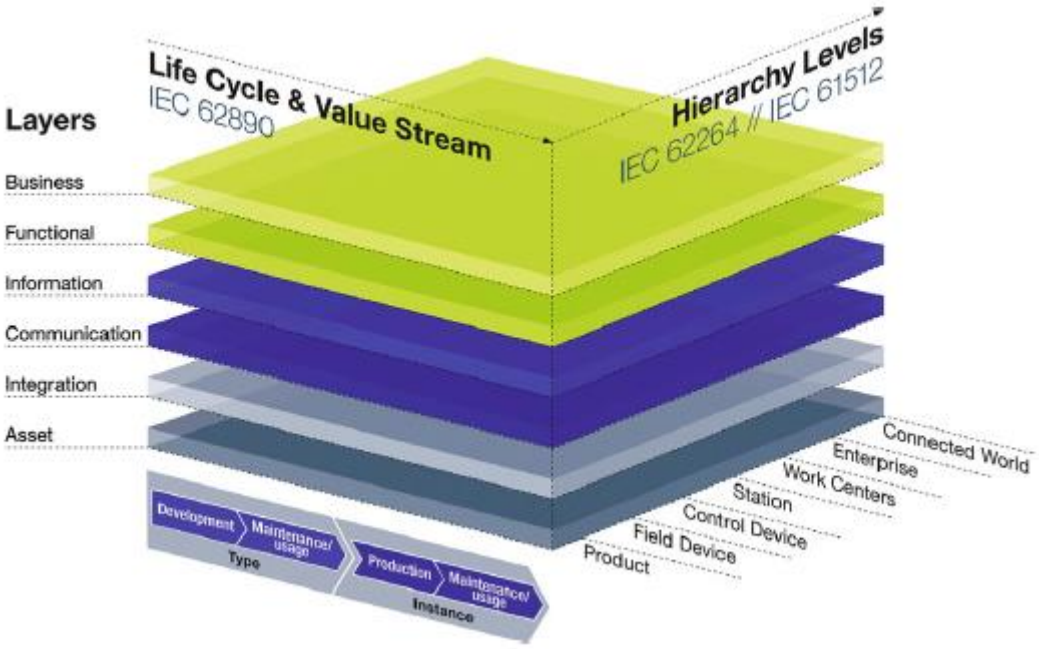

Figure 19: Industry 4.0 reference model RAMI4.0 [60], [64], [65]

4.4 Internet-of-Things (IoT)

The system type focussed on in this study is IoT systems. IoT systems are used in a fast-growing industry and are present in many everyday items and systems. As the complexity of IoT systems is increasing, usability and proper system analysis will play an increasing role in IoT system development.

According to Atzori [66], the reason for the lack of a clear and standard definition for IoT is the fact that the name "Internet-of-Things" is the combination of two names. This inspires researchers to lean more towards either the "internet", or the "things" side of the IoT spectrum. This creates two different types of definitions for IoT. Hence no single standard definition currently exists.

The IoT paradigm can be seen as a convergence between three perspectives with different outlooks. These outlooks are illustrated in Figure 20. The three perspectives are things, semantic, and internet-orientated. When things and internet perspectives converge, elements like sensors can talk to each other over the internet. When the internet and semantic perspectives (or “visions”) converge, internet applications with data analysis capabilities are created. [66]. These three visions (or “perspectives”) focus mainly on the technology side of IoT, as with IoT views in general. This adds to the research challenge stating that the focus is not on all stakeholders and the whole system life cycle.

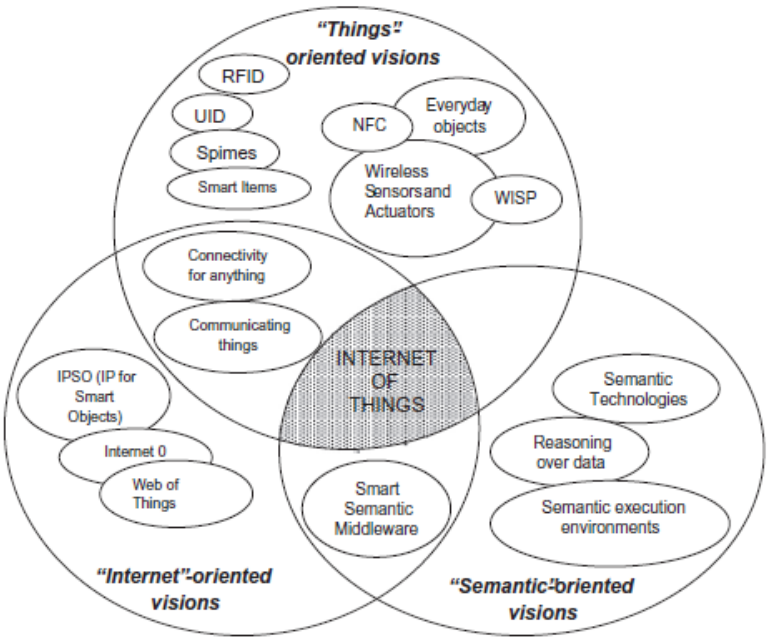

Figure 20: IoT paradigm converging visions [66]

Some challenges of IoT are internet availability, security issues, cost-efficient designs, energy consumption and output, computational capabilities, scalability, lack of employees' skills and knowledge, error tolerance, and user-acceptance [67]. These are all important challenges that need to be met for an IoT system to function properly. Functional challenges such as computational abilities are often met in the design process, but usability challenges such as user-acceptance are not always met. A lack of attention to usability factors may lead to malfunctioning or hard to use systems.

There are some gaps evident in IoT, for instance, the lack of exploitation of information through the system life cycle, the presence of a gap in the connection of "things" like sensors, and data analysis, as well as a lack of using the information throughout the whole system life cycle [68]. This is the reason why usability in the full system life cycle has become important to this research. Some existing models and frameworks focus on IoT, either in use or just theoretical.

- Atzori [66] and Yang [63] defined models for more traditional IoT frameworks. These models focus more on technology and information.
- Chan [69] defined an IoT business model that focuses on technology and end-user.
- Kiritsis [68] developed the closed-loop system life cycle management (C-PLM) perspective for intelligent products. It focusses on the system information.
- Atzori [70] also developed social IoT architecture. It focusses on the social relationships and interaction between objects.
- Peschke [71] developed the PABADIS'PROMISE with the focus is on system control.

None of the IoT frameworks has a focus on usability, and only the C-PLM for intelligent products represent the whole system life cycle.

The research indicates a lack of usability in IoT. The research also indicates a lack of a general definition of IoT [66], but researching IoT characteristics do contribute to giving a general definition of IoT. In designing IoT systems, the focus tends to be on the end-user. This is also shown in the list of IoT challenges listed earlier [67]. Most of the challenges affect the end-user the most. More attention should be given to challenges regarding the whole system life cycle as all aspects of usability must be considered.

4.5 Ergonomics

Ergonomics is a growing field in product design. Ergonomics and usability are closely related and shares characteristics. It is a well-studied field, and there are many existing models and frameworks.

The International Ergonomics Association defines ergonomics as [72]:

> "Ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance."

Professionals who work in the field of ergonomics collaborate in the design, implementation and evaluation of tasks, products, jobs, environments and systems. Their main goal is to create an environment that suit peoples' needs, abilities and limitations [72]. Ergonomics is a holistic approach to improve physical, cognitive, social, organisational, and environmental factors, etc. [72].

Ergonomics are also called human-factor engineering (HFE). Human-factors depend on the environment. These environments consist of a complex combination of a physical, organisational and social environment. The combination of these three environments creates systems and the components in these environments can be dependent on each other [50]. Many factors influence ergonomics. Some factors are [50]:

- Global change of work systems;
- Cultural diversity;
- Information and communication technology;
- Enhanced competitiveness and the need for innovation; and
- Sustainability and corporate social responsibility.

These factors can have a significant influence on ergonomics and system design. For example, South Africa's workforce is often faced with health issues like HIV, respiratory diseases, cardiovascular diseases and other infectious diseases. Intended violence, an unstable political environment and workers' unions are also factors to keep in mind. The working capacity of the population is thus influenced by these factors and affect the decisions of an ergonomist. Ergonomists in countries like this have the potential to improve the performance and well-being of the workforce involved in their projects [50].

Existing models and frameworks focusing on ergonomics, either in use or just theoretical:

- Dul [73] defined the ergonomics in company strategies where the focus is on manufacturing and the end-user.
- Kleiner and Karsh developed the macro-ergonomics framework [74], [75], where the focus is on organisational factors.
- Karsh also developed micro-ergonomics [75] and meso-ergonomics [75]. Micro-ergonomics focusses on technical and individual factors, and meso-ergonomics focus on multi-level relationships.
- Haines developed and validated participatory ergonomics [76]. It focusses on the management of people.

None of the existing models and frameworks represents the whole system’s life cycle and usability analysis.

Ergonomics is still a factor in usability and adds to defining usability in IoT systems. Ergonomics is not the focus of this study, as it is already implemented in engineering systems. Ergonomics principles indicate that all life cycle phases aren’t factored in. There is also a focus on the end-users. Ergonomics principles can thus be extended to all stakeholders.

4.5.1 User interface design principles

When designing a user interface (UI), there are sources with UI design principles that give guidelines for designing such an interface. UI design principles are implemented in the requirements and design phases. Analysing UI design principles provides information for usability principles.

Some UI design principles sources are:

- IBM CUA: Guide to user interface design [77], [78];
- The Windows interface: An application design guide [79];
- ISO 9241-14: Menu dialogues [80];
- ISO/IEC 11581: Icon symbols and functions [81], [82];
- Macintosh human interface guidelines [83]; and
- Interaction Design Foundation: User Interface (UI) Design [84].

An important aspect of usability is the positioning of user interface information on the physical layout/design. Using eye-tracking research, researchers identified different patterns users use when reading user interfaces like web pages, but these principles can be used in other user interfaces as well. A discussion on an important aspect is given below, specifically because it will be implemented in the pivot interface design.

One of the patterns is in the shape of the letter F or E. With the F shape; users start reading horizontally across the upper part of the text. Then they move down a bit and start another horizontal movement, usually a shorter area. Then their eyes move downwards, scanning headings or keywords in a vertical movement. This means that users read more at the top of the page and then the first few words of the lines below the horizontal eye movements. These eye movements are usually in the content area and not the navigation bars as some would suggest. For people reading right-to-left, like with Arabic, the F shape is flipped [85].

The F-pattern is not the only scanning pattern, but it is still a powerful scanning pattern to consider. Due to the user trying to be efficient and not wanting to read everything, users tend to scan in patterns over the page or screen. Not abiding by reading patterns may cause users to miss important information [85].

The designer must design the interface for leading the user's eyes to the important information. This can be done by [85]:

- Including a good summary or at least the most important information in the first few paragraphs (preferably the first two paragraphs);
- Headings and subheading can help a user to scan through the content to lead the user to relevant information;
- Ensure that the essential information is in the first few words of the headings;
- Make important words bold;
- Bullets and numbers are eye-catching and will give information of lists, processes or procedures best.

Some other guidelines for user interface design are:

- Make sure important elements are easily discoverable [84];
- Use appropriate spacing and positioning (don’t underestimate the power of white space) [79];
- Incorporate clear and descriptive messages to lead the user [78];
- Don't use too many layers with a heading, preferably no more than three [86];
- Keep the interface simple with a minimalist design [84];
- Use the WYSIWYG (what you see is what you get) principle to minimise hidden features [83];
- Use colour, contrast, font sizes, bold or italic typewriting, or underlining to guide the user to relevant information or elements [84];
- Group elements intuitively together to simplify scanning [79];
- Ensure that the interface is responsive with minimal waiting times [78];
- Give appropriate feedback and dialogues [83];
- Use bread crumbs to simplify navigation [86];
- Keep the number of actions to the minimum when performing a task, as well as navigating to the appropriate setting/page/element [84];
- Draw the user’s attention to important buttons or menus [79];
- Ensure that the user can solve errors quickly and efficiently [83];
- Use consistency to your advantage [84]; and
- Use colours, icons, familiar buttons and language, and appropriate dialogues to improve intuitiveness [79].

UI design principles are implemented in the usability framework. The focus is also on the end-user and the end-product. The principles are implemented in the pivot controller user interfaces (case study).

4.6 Centre pivot irrigation systems

A centre pivot irrigation system (CPIS) is a low to medium pressure irrigation system that is fully mechanised and automated [87]. This permanent assembly is used in farming and consists of a pivot structure and controller. The system may include probes in the ground to monitor water and fertiliser levels to prevent under- or overirrigation. The probes send the information to the control system, and the control system automatically adjusts their movements. The system can also be controlled remotely by a smartphone and computer application [88], [89].

Figure 21a shows a centre pivot (CP) with a control panel. If remote control is present (via smartphone or computer), the information is sent from the smart device to the control panel [87], [88], or inputted into the control panel itself. It is then sent to the pivot structure, and adjustments are made, or the command is executed. If the remote control is not present, the command and adjustments are entered into the control panel itself. Figure 21b shows the structure when working. The water is delivered over a large horizontal space.

Figure 21: a) Centre pivot with a control panel, and b) Pivot irrigation structure irrigating [88]

Figure 22 shows a diagram of the basic components of a CPIS. The centre of the pivot field is called the pivot location. This is where the pivot mechanism and command area are located. The pivot mechanism, central tower, or just pivot as it is called [87], [90], [91] is an “A” shaped steel structure where the control equipment like the pivot control panel is attached. The pivot is fastened to the ground with a concrete base. The fixed input supply of water (hydrant) is also located at the pivot. The water inlet needs to be al least 0.65 bars for functioning efficiently with Low Energy Precision Application (LEPA). 0.65 bar is needed at the end of the pipeline for proper functioning, and the CP has an inlet of almost 3 bar for normal functioning [87].

The control panel controls the flow rate by adjusting the speed of the CPIS. Other features are also included in the control panel. The control panel is also used for diagnostics and contains sensors for voltage, pressure, etc. LED lights help with diagnostics. The control panel also includes automatic features like an automated starter, shut-off, position stop device, and hour counter. The control panel is usually available in various levels of control [87].

From the pivot, the series of spans [90] or pipeline is the pipes and cables that run from the pivot, with the sprayers or sprinklers attached to the pipe moving the water. Flexible joints are located at the ends of each span for efficient pipeline movement. This includes side-to-side, as well as up and down movements, according to the field typography [87].

The sprinklers vary in size and increases in size to the outside of the field. This is to increase the flow rate to the outside of the field to account for the larger diameter. The sprinklers are

used in LEPA mode. For efficient irrigation, overlapping is important, as well as to prevent the wheels from getting stuck from overirrigation in the tyre tracks. Part circle sprayers are used for the sprinklers closest to the towers [87].

The pipeline rotates around the pivot in a clockwise (forward) or anti-clockwise (reverse) movement, as indicated with the travel direction arrow (illustrates forward movement) (Figure 22) [87].

The pipeline is supported on towers (also called drive units [90]). The towers rest on wheels with the mechanism (tower box [90]) moving the tower in the travel direction. From the last tower, the pipeline overhangs for a distance with sprinklers. At the end of the span, an end gun, also called a sprinkler gun, can be attached to extend the reach of the CPIS. The end gun can be turned on at corners and other locations to reach the full extent of the field [87].

An automated alignment system is used to ensure that the pipeline is straight. If the alignment system determines that one or more towers are not in position, the pivot controller stops the drive system. When other variables like pressure and voltage are outside the prescribed limits, the drive system is also stopped by the control panel [87], [92].

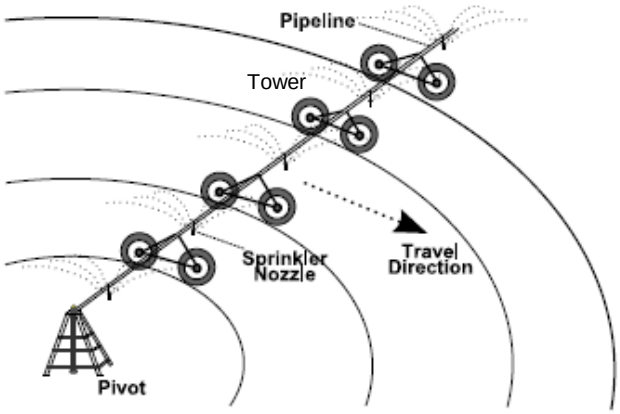

Figure 22: Basic centre pivot system components [91]

There are existing pivot irrigation companies in South Africa. They are Senter360 [88], Valley ([93], [94], [95], [96]), Talbert [97], Agrico ([98], [99]), Zimmatic [100]–[102], and Automation Products ([103], [104], [105], [106], [107], [108], [109], [110], [111]). The available options for each brand vary from basic to advanced options, with the available functions also varying.

These existing CPIS systems provide guidelines and generalised functions used in the CPIS design in Section 5.5.

4.7 Synthesis from literature

4.7.1 IoT definition

From the literature, a definition of IoT is created, which includes the full life cycle and all stakeholders. This definition of IoT forms the foundation for the usability framework.

As seen in Section 4.4, multiple definitions for IoT exist in literature. Abdel-Basset [112] defined IoT as:

> "We can define it as a set of physical and virtual objects which are connected via a network for communication and sensing or interaction with the internal and external environment."

Kiritsis [68] defined IoT as:

> “A global network infrastructure where physical and virtual objects with unique ID are discovered and integrated seamlessly (taking into account security and privacy issues) in the associated information network where they are able to offer and receive services, which are elements of business processes defined in the environment they become active.”

These definitions illustrate a view of IoT, limited to the electronics like sensors and actuators. In this study, the IoT view is expanded to encapsulate the entire system surrounding the IoT electronic devices. The entire system can be seen as IoT.

The IoT environment used in this study exists of three main environments, as illustrated in Figure 23. The largest environment is the development environment, where development usability is mainly located and implemented through the management and strategies for the usability process. The developmental environment includes the development team, client, end-user environment, business environment, etc.

The second environment is the system environment. This is the environment in which the system operates. It includes the end-user environment. The third circle is the product itself. The product is the centre of the whole system. Product usability is located in this circle. The usability in this circle is mainly regarding the product and its operations.

Understanding the engineering system environment will provide a foundation for the new definition for IoT. It also indicates the environments where usability will be implemented in the usability framework.

Drawing from literature the following proposed definition for an IoT system towards solving the research challenge is: An IoT system is a system that exists in the 1) development, 2) system and 3) product environments, where the system consists of five layers, namely: 1) business, 2) information, 3) operations, 4) technical, and 5) detail layers. IoT includes users and operations over the full life cycle. The system users are categorised as system 1) actors, 2) experts, 3) decision-makers, and 4) influencers.

The proposed IoT concept is explained in depth and fully in the rest of Section 4.7.

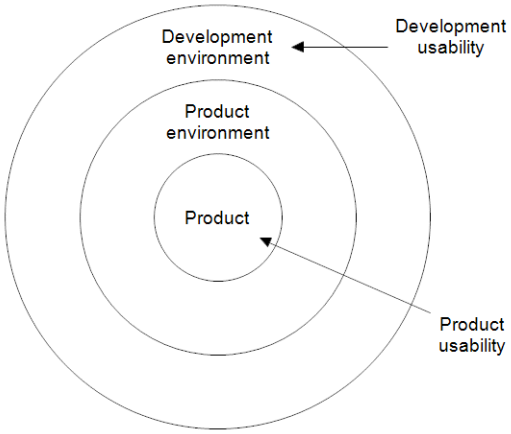

Figure 23: IoT and engineering system environment

IoT consists of five layers, as shown in Table 3 [113]. These layers are the business layer, the information layer, operations layer, technical layer, and detail layer. More information is given about each layer in Table 3 and Sections 4.7.1.1 to 4.7.1.5.

From the detail layer upwards to the business layer, the maturity and IoT applicability or “readiness” increase. IoT can only be efficiently implemented if all layers are involved [113].

Domain speciality increases from the information layer downwards to the detail layer. The specialist knowledge needed in each layer increases, such as data management, data mining and the specialities of the development team and design engineers [113].

Data science also increases from the information layer to the business layer. Data management is important in both the information and business layers. Business needs to be supported by data analysis as business information is one of the most important cornerstones of the decision-making process [113].

Both top-down and bottom-up approaches are applied in the IoT system. From the detail layer, data and information, processes, and infrastructure are implemented upwards through the

layers. Information and data are also implemented from the top downwards through business and strategic decisions [113].

The layers form the majority of the IoT definition, where the system stakeholders and users can be derived from the layers and system architectures.

Table 3: IoT layers [113]

IoT layer	Main focus	Information / intelligence	Timescale	Process
Business layer	Optimisation	Strategic / system intelligence	Long term	Optimisation processes
Information layer	Contextualisation	Management / system information	Medium term	Management processes
Operations layer	Application	Core process / product information	Short term	Core operations processes
Technical layer	Infrastructure	Machine / materials information	“Real-time”	Technical processes
Detail layer	Components	Data / technical information	”Real-time”	Software coding etc.

4.7.1.1 Business layer

The business layer contains the business aspects and upper company levels. The main focus of the business layer is business and system optimisation. Strategies and policies contribute to the flow of information and work that needs to be optimised for efficacy. These policies are implemented throughout the entire IoT system for optimisation and include business, process optimisation, value chain engineering, compliance, quality optimisation, enterprise risk management, market intelligence, and ERP [113].

Risks in the business layer include competitive lag, legal risk, compliance, reputational damage, and shareholder loss. The benefits in the business layer when the IoT system is implemented efficiently are innovation, risk management, optimal systems, competitive intelligence, and quality [113].

4.7.1.2 Information layer

The information layer contains most of the data analysis and data mining. This includes data scientists, informatics specialists, etc. Data is analysed and given context through the data analysis and visualisation process [113].

Tasks occurring in the technical layer are analysing existing infrastructure, identifying technology shortfalls, risk indicators, and defining technical performance indicators, establishing informativity, adjusting infrastructure, monitoring equipment, materials, work, and control, optimising equipment, methods, and tools [113].

Risks in the information layer include service and/or product quality failure, productivity loss, relationship breakdown, and contractual failure. The benefits in the information layer when the IoT system is implemented efficiently are life cycle information flow, efficacy data, service data, quality data (variability), and CRM information [113].

4.7.1.3 Operations layer

The operations layer contains the engineering system life cycle operations. This includes operations domain specialists, technicians, field personnel, etc. The main focus of the information layer is the application of procedures. This means that operations and policies are implemented and applied to the system life cycle [113].

Processes are implemented throughout the entire IoT system for optimisation and include logistics, centralised control, enabling processes, local IT networks, methods and techniques, people, and procedures [113].

Tasks occurring in the operations layer are analysing operational processes and systems, identifying operational performance indicators, operational opportunities and operational risks, adjusting IT systems (Lean construction), visualisation of operational and infrastructure data, optimising processing, and infrastructure [113].

Risks in the operations layer include people failure, production downtime, procedural failure, and availability. The benefits in the operations layer when the IoT system is implemented efficiently are process performance data, asset data, materials data, and product quality data [113].

4.7.1.4 Technical layer

The technical layer contains the engineering system life cycle equipment, technical infrastructure, and products [113].

Processes are implemented throughout the whole IoT system for optimisation and include materials, local control, enabling processes, production networks, sensors and actuators, machines and tools, and utilities [113].

Tasks occurring in the technical layer are analysing existing infrastructure, identifying technology shortfalls, risk indicators, and defining technical performance indicators, establishing informativity, adjusting infrastructure, monitoring equipment, materials, work, and control, optimising equipment, methods, and tools [113].

Risks in the technical layer include machine failure, utility failure, materials failure, control failure, and industrial network failure. The benefits in the technical layer when the IoT system is implemented efficiently are materials data, tracking data, operator performance data, machine performance data, and network data [113].

4.7.1.5 Detail layer

The detail layer contains the engineering system life cycle requirements, specifications, designs and development data. The main focus of the technical layer is design. This means that requirements, specifications, designs, and development data are created and applied to the engineering system life cycle [113].

Processes are implemented throughout the entire IoT system for optimisation and include technical risk, requirements analysis, systems engineering management plan (SEMP), specifications, and planning. Tasks occurring in the detail layer are analysing system requirements and specifications, technical risks, creating the work breakdown structure (WBS) and IoT strategy, and optimising system design [113].

Risks in the detail layer include design flaws, design incompatibility, incorrect assumptions, client-developer relationship failure, unidentified risks, and inadequate implementation data. The benefits in the detail layer when the IoT system is implemented efficiently are competitive design, risk management efficacy, and implementation efficacy [113].

4.7.1.6 IoT speciality requirements

For an accurate IoT definition, it is important to identify the system users and stakeholders. The four main groups of human resources directly and indirectly involved in IoT systems are system actors, system experts, system decision-makers, and system influencers [50]. As the study focuses on technical aspects, only a few specific human resources are discussed in this section. They are [50]:

- **System actors:** e.g. skilled technicians such as manufacturing personnel, field support personnel, installers, and dismantling personnel;
- **System experts:** e.g. systems engineer, IT engineer, data scientist, informatics specialist, operational domain specialist;
- **System decision-makers:** e.g. business domain specialist.

The specialists mentioned in this study are mostly domain specialists. The respective fields of the domain specialists may vary from electronics, telecommunication, agriculture engineers, etc.

Systems engineers are the interfaces between the client and the development team, which ensure that communication and work flow efficiently. They are responsible for specialised operational modelling, including functional diagrams that are used for verification, validation, context, and design guidance. Systems engineers are also responsible for engineering management and specialised system optimisation.

IT engineers are responsible for specialised IT system design, IT engineering management, and specialised network optimisation. This includes software development and connectivity.

Data scientists are responsible for data contextualisation, specialised cyber modelling, and specialised data analysis. Data science specialists generally specialise in data management and processing. They receive data and cleans the data up for the informatics specialist and other departments.

Informatics specialists are responsible for information contextualisation, specialised information visualisation, and specialised information management. Informatics specialists use the information from the data scientist. They are specialists who specialise in the visualisation and management of information.

Operational domain specialists are responsible for operational contextualisation, process and resource knowledge, and experience. Generally, the operational domain specialists manage operations, processes, and the resources accompanying them.

Business domain specialists are responsible for business intelligence contextualisation, business value definition, and business knowledge and experience. These specialists also include business analysts.

For the usability framework, the users and stakeholders in the system need to be identified to extend the IoT definition to all users and stakeholders.

4.7.2 Usability definition

To form a definition of usability in a full life cycle system, the properties of usabilities need to be analysed. The definition of usability is used in the usability framework. Understanding usability requirements in all the life cycle phases is also key to the usability framework.

The current official definition for usability by the ISO 9241-11 for user interface usability is: “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [10], [11].

This definition focusses on the end-user, as well as defined usability heuristics, as defined by Nielsen [12]–[14], Shackel [15], Shneiderman [16], [17], Norman [17], etc. In this study, the definition of usability is extended to the entire systems engineering life cycle and includes all users and stakeholders.

4.8 Summary

The literature study is summarised in Table 4. The arrow pointing upwards (↑) indicates that the focus area validates the research challenge in that column. The arrow pointing downwards (↓) indicates that the focus area is used in the synthesis of the research solution in that column.

Table 4: Literature study validating research challenges and solutions

Focus areas - literature	**Research challenges**	**Lack of information on usability in IoT**	**No clear definition of IoT in general**	**Usability over full life cycle in IoT systems not defined**	**Usability in irrigation pivot controllers not fully applied**	**Focusses mostly on end-user usability**
4.1 Systems engineering				↓	↓	↓
4.2 Industry 4.0			↑↓	↑↓		
4.3 IoT		↑	↑↓	↑		↑
4.4 Usability (and user interface design)		↑↓		↑↓	↓	↑↓
4.5 Ergonomics		↑		↑↓		↑↓
4.6 Centre pivot irrigation systems				↓	↓	↑
Literature focus areas	**Research solutions**	**Create framework for usability in IoT systems**	**Give clearer definition of IoT**	**Create framework for usability over full life cycle**	**Usability applied to irrigation pivot controller - ADR case study**	**Expand focus to all IoT system stake-holders**
Detailed solution						

5 SYNTHESIS OF A FRAMEWORK FOR USABILITY IN THE FULL SYSTEM LIFE CYCLE

An important consideration, considering the sustainability of a system in general, is how usability influences the human-system interaction over all phases of a full life cycle. If the product or system under development is the end goal, and all tasks towards that goal are not fully “usable”, the resulting product may not be optimally usable. Furthermore, should future generations need to upgrade or change the system, such changes may be sub-optimal, if not near impossible in cases where design decisions and other important meta-information are not available. Hence, it is necessary to consider a general system’s usability over all system life cycle phases to understand the full impact of usability on a system.

This chapter covers the creation of a usability framework for IoT systems, across a system’s full life cycle, and including all users and stakeholders over the full life cycle. Such a framework will provide an appreciation for the value of usability and provide key considerations when developing a new IoT system in the 4th and 5th industrial revolutions.

The usability framework consists of the 1) full life cycle phases, as identified in Section 5.1.1, 2) system users and documentation per life cycle phase, as identified in Section 5.1.2, and 3) usability heuristics, as identified in Section 5.2. For each life cycle phase, the usability heuristics are applied, with consideration to the users and documentation identified for that phase.

5.1 Generalised system analysis

The properties of general IoT systems and the IoT system life cycle need to be analysed to create a full life cycle usability framework. The general IoT system life cycle was obtained from literature (Section 4.2.2), and the higher levels of the system are discussed in this chapter.

The functional analysis of a general IoT system is divided into:

- The functional flow on an operational level down to the equipment level; and
- System architectures.

When considering system full life cycles in this research, the analysis of usability is done at the operational level as development activities involving humans take place at this level. Product usability aspects are considered inside of design activities, for example, and are not specifically included in this analysis as these have been addressed in other texts [12], [14], [21].

Figure 24 shows a high-level operational architectural diagram of a general IoT system. Only one end-user is shown in the diagram, but multiple end-users may be present in the system. The diagram shows the variety of users and stakeholders in a general IoT system. Each user and stakeholder should be factored into the requirements, specifications, designs, etc. in ways that will be analysed below.

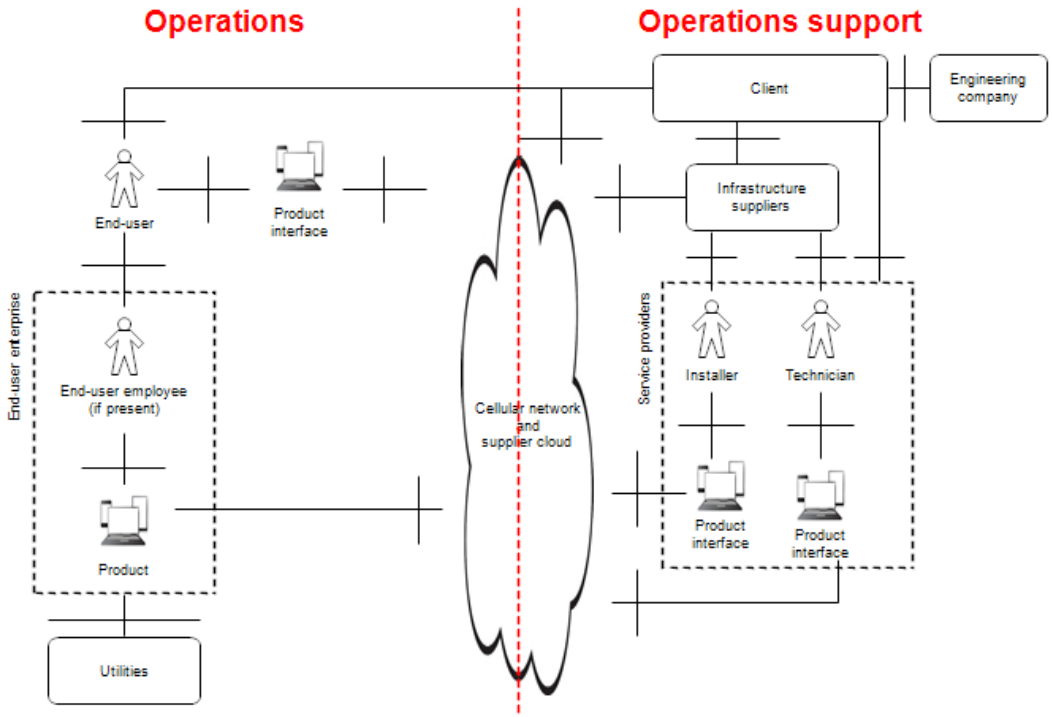

Figure 24: High-level operational architecture

5.1.1 General system life cycle

The system life cycle operational level functional flow is shown in Figure 25. The acquisition phase is when the product is developed, whereas the utilisation phase is when the product is manufactured, installed, operated, and maintained.

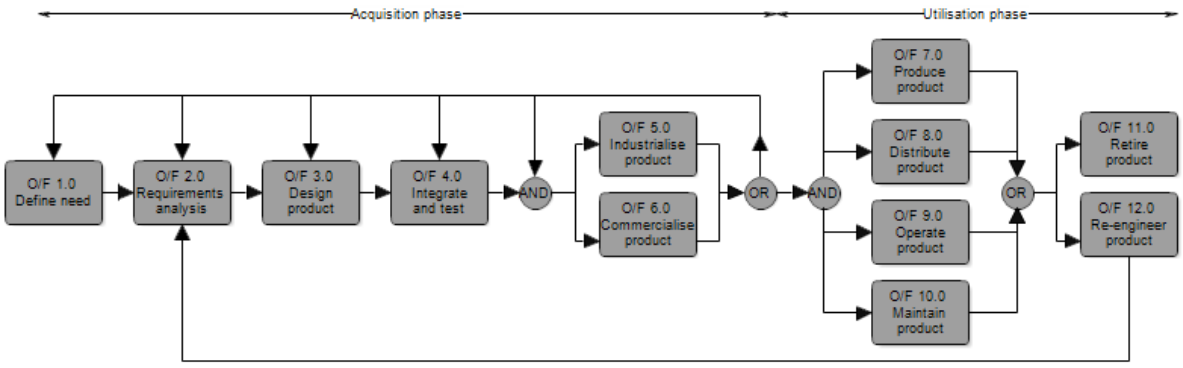

Figure 25: System life cycle operational level functional flow

The Definition of Need phase is included in the usability analysis, as the requirements flowing from this phase will be considered in the following phases – the first two phases are combined below from a requirements perspective. The Requirements Analysis phase (O/F2.0) sets the foundation of a product or system. When requirements are not specified correctly, designs in all the following phases will be affected, and ineffective equipment will result from a usability perspective.

The system life cycle was divided into five different phase groups, as discussed in Section 4.1.4. Figure 26 shows the grouping of the phases.

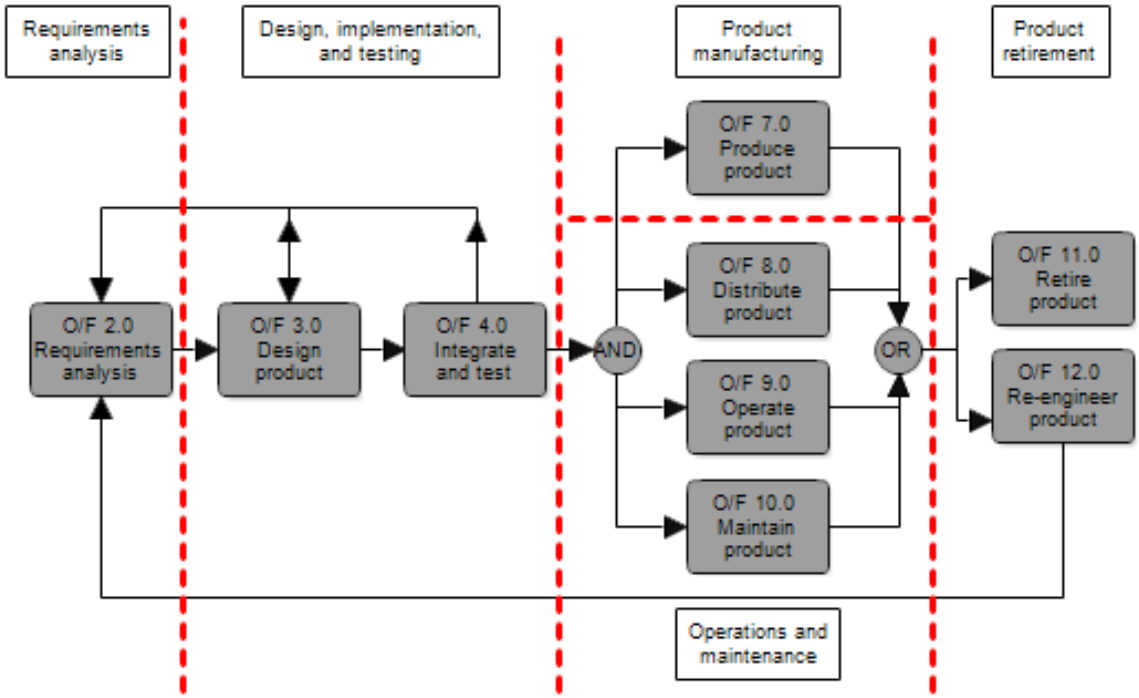

Figure 26: System life cycle as divided into groups

Human resources (Section 4.7.1.6), including personnel, independent contractors, clients and/or end-users, etc. are present in an IoT system. Interfaces between humans and technical elements of the system (functional and non-functional) are important as these determine the perspectives of usability in a system context. In the following sections, different system architectures, aligned with the system life cycle phases above, are considered, and interfaces between different elements are discussed.

5.1.2 General system architecture

The architectures for the five different phase groups identified in Section 5.1.1 are analysed in terms of their respective resources in this section. These phases are (i) Requirements

Analysis, (ii) Design, Testing, and Implementation, (iii) Product Manufacturing, (iv) Operations and Maintenance, and (v) Product Retirement.

With IoT systems, one part of the system is the equipment like the electronics-, sensors-, telecommunication systems, etc. The other part is the human resources, documentation, data, and information of the system.

The **Requirements Analysis** architecture is seen in Figure 27. The client may be part of a company, or it may be the end-user. Domain specialists provide information regarding the infrastructure and equipment other from the engineering company equipment. The domain specialists may be contracted or employed in the client company.

The systems engineer is the interface between the client and the development team and needs to make use of both the client and the development team ontologies to form this interface. The systems engineer must document system requirements in a format and language the development team would understand.

The operational domain specialist understands the client operations and ontology and is consulted by the systems engineer to complete the functional analysis at the operational level and to define system requirements.

System requirements define the total operational definition of the system under development and are under the control of the systems engineer, who interfaces to all role players.

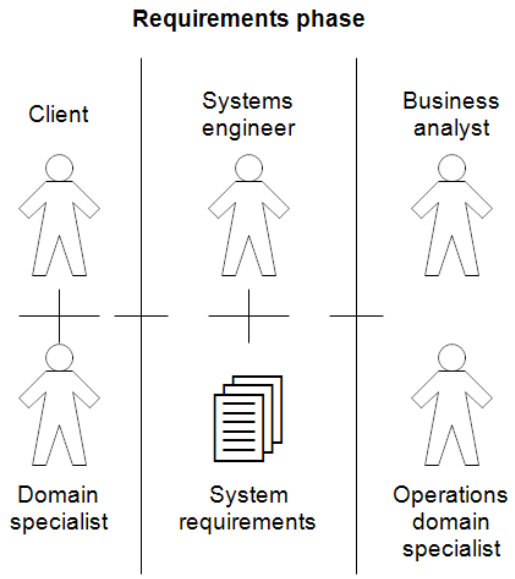

Figure 27: Requirements analysis architecture

The **Design, Implementation, and Testing** architecture is shown in Figure 28. Again, a domain specialist ensures the validity of operational requirements, the client remains present, and the systems engineer ensures client requirements are managed. The development team should comprise of IT engineering resources as IoT infrastructure relies upon IT systems and data science resources to ensure massive amounts of data can be analysed, structured and presented to information specialist resources that will structure and present data for management purposes. Design engineers are present to ensure a structured approach is followed in line with requirements (and to form an interface with the systems engineer).

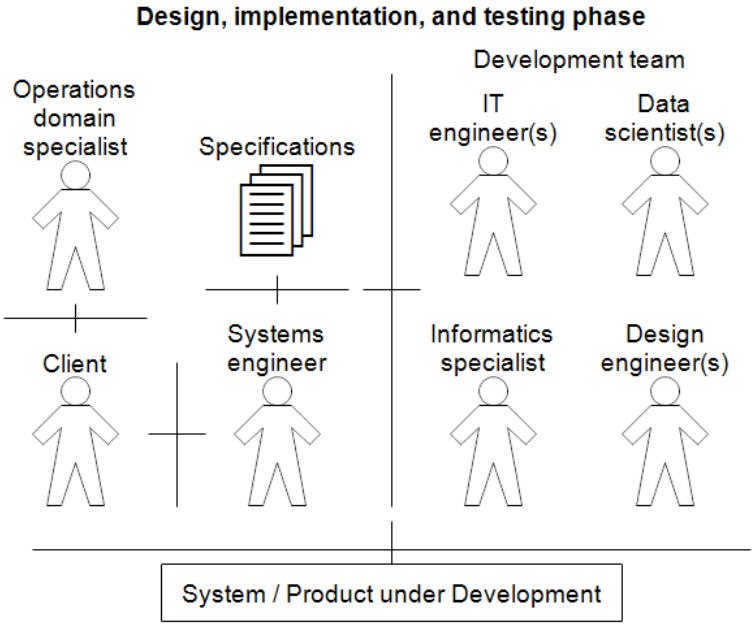

Figure 28: Design, implementation, and testing architecture

The **Product Manufacturing** architecture is shown in Figure 29. Manufacturers use manufacturing data packs to define production and quality assurance processes, procedures, and methods. The systems engineer ensures a manufacturing datapack is compiled by design engineers (often specialists in production systems) using design information and manufacturing specifications. The manufacturers obtain information from production specialists that ensure quality.

In some cases, manufacturing and quality control support systems are developed *after* the Design, Integration, and Testing phase has been completed. This may lead to a lack of usability in product manufacturing that must be addressed by proactive design.

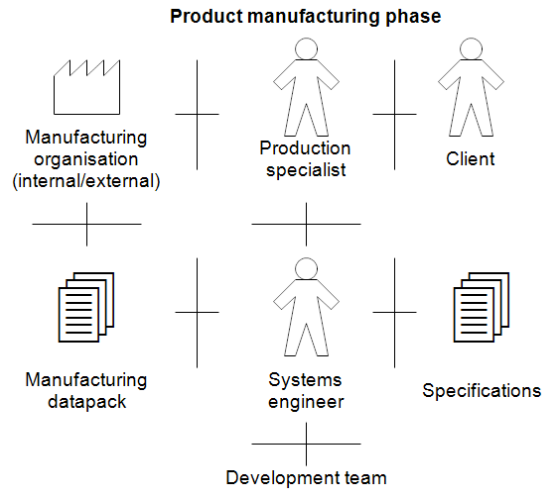

Figure 29: Product manufacturing architecture

The **Operations and Maintenance** architecture is shown in Figure 30. Technical information is communicated to the end-user via a user manual, online support systems, and training.

From literature, it became evident that usability is mostly applied to the end-product itself as will be discussed in the case study in Section 5.5.

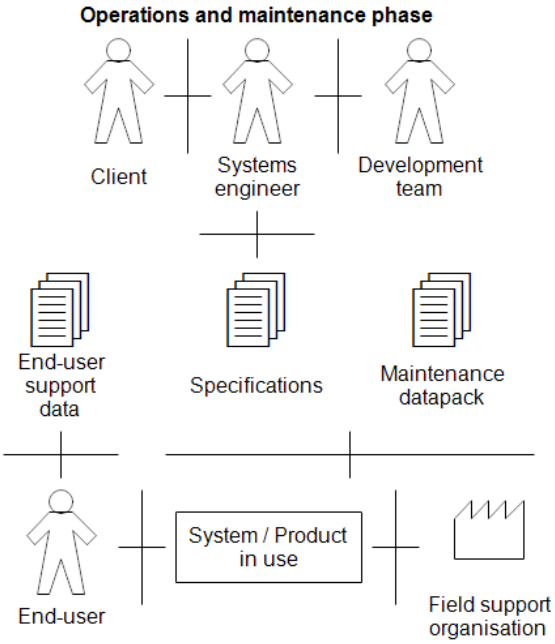

Figure 30: Operations and maintenance architecture

The **System Retirement** architecture is shown in Figure 31. The knowledge in this group phase includes technical and materials knowledge. The documentation indicates what parts and components are present, as well as their disposal and recycling requirements.

When product support is discontinued, the product owner may choose to retire the product. Organisations specialising in disposing and recycling system parts and components are typically contracted. Often, a product is simply re-engineered and reintroduced into operations after following the same path as a new product, but with reviewed as opposed to new requirements.

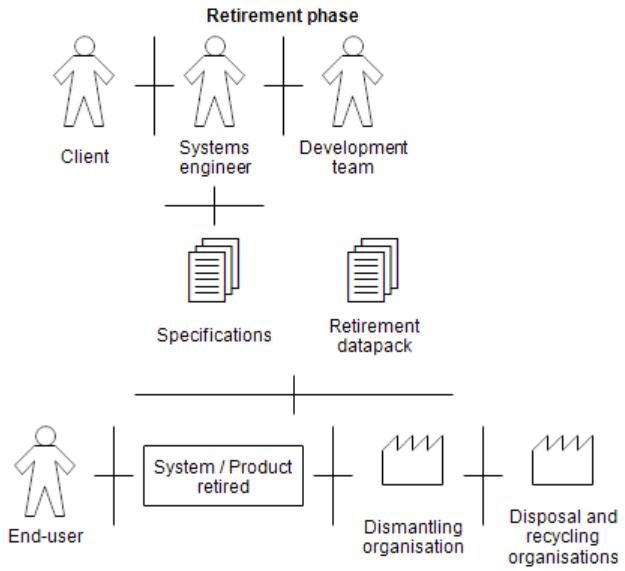

Figure 31: System retirement architecture

By considering all interfaces in the system architecture across the full life cycle, it is possible to define a usability framework that is more comprehensive than just end-product usability. This will, overall, make a product or system more supportable over its full life cycle.

5.2 Generalised life cycle usability heuristics

As from the literature study, Nielsen's ten usability heuristics [12]–[14] form the main reference for usability heuristics as applied in this section. Nielsen's heuristics are used for product design, while a more generalised set of heuristics is required for the full life cycle. Generalised heuristics will be obtained by converting each Nielsen heuristic (applied to the end product) to a general heuristic (applied to the system life cycle). Each of the resulting heuristics, presented in pairs, will be referred to as an Information Systems heuristic (from Nielsen) or IS heuristic for short, and a generalised heuristic that applies to the full life cycle process.

Section 5.2.1 to 5.2.10 thus present and compare two sets of usability heuristics, namely the original Nielsen heuristics and the generalised heuristics, which will also be referred to as general life cycle heuristics. In the following sections, Nielsen's heuristic is provided for all ten heuristics, with the generalised heuristic given below Nielsen's heuristic to simplify reading.

5.2.1 Heuristic 1

Nielsen: Visibility of system status.

General: System life cycle representation and status.

5.2.1.1 IS perspective:

End-users should be able to see the current state of the system, including specific error and operational states. Efficient and appropriate program interfaces are designed and implemented to achieve this heuristic goal and can be seen as a general operational function.

5.2.1.2 General life cycle perspective:

A representation of the developmental state of the system should be available to appropriate development team members, business associates, and the client. Documents like progress reports, as well as representative models, prototypes, etc. are used for these representations.

Specifications are also needed for definition, verification, and validation purposes. Good quality specifications ensure an accurate representation of the system. End-users still need to be able to access the system status visually from the product through the manufacturing phase onwards.

5.2.2 Heuristic 2

Nielsen: Match between system and the real world.

General: Contextualisation and accuracy of real-world system representation.

5.2.2.1 IS perspective:

Decision support and task execution should be executed efficiently and accurately. A realistic representation and linking the system to a real-world application minimises the chance for confusion for end-users and other stakeholders.

5.2.2.2 General life cycle perspective:

A link to the real world is essential for systems diagrams and is achieved by creating realistic and accurate diagrams derived from the functional analysis. Realistic system representations are crucial during the during system acquisition phases to ensure accurate and efficient product designs. A realistic representation should enable system stakeholders with sufficient decision and development support.

Documents, diagrams, and models must contain sufficient context. This also requires the use of the appropriate ontologies.

5.2.3 Heuristic 3

Nielsen: User control and freedom.

General:

- System life cycle quality,
- Requirements management.

5.2.3.1 IS perspective:

A user should be able to interact with the product and manage the settings, configuration, input values, etc. This provides users and stakeholders with a minimum required level of control.

5.2.3.2 General life cycle perspective:

The client/end-user should have control over system requirements. The systems engineer and development team, however, control the requirements elicitation process and manage technical requirements.

The voice of the customer is thus heard through requirements elicitation and controlled involvement in requirements management [114].

5.2.4 Heuristic 4

Nielsen and general: Consistency and standards.

5.2.4.1 IS perspective:

Consistency is achieved by using *de facto*, formal standards, as well as usability patterns in the development process of a product. These standards include, menu dialogues [80], F-shaped reading patterns (Section 4.5.1) [85], icon design [81], [82], use of colours, and others.

5.2.4.2 General life cycle perspective:

When creating representative system diagrams and documentation, standard methods and techniques should be used. This includes a system functional analysis, where standard functional flows and architecture models are used.

Standard methods and tools for documentation, development, production, etc. should be used. Product standards in the industry and consistency with other product families should be fulfilled.

5.2.5 Heuristic 5

Nielsen: Error prevention.

General: Technical risk management.

5.2.5.1 IS perspective:

End-user and product interactions should be carefully planned and automated as much as possible. This is to limit possible user errors employing comprehensive design for error prevention using appropriate techniques.

5.2.5.2 General life cycle perspective:

Risk management should be used to identify and manage potential errors, failures, and other issues during development. Risk management techniques include techniques like failure mode and effect analysis (FMEA) and failure mode, effects, and criticality analysis (FMECA). During development, design, implementation, testing and production failures must be considered in conjunction with the design for the robustness of the target system.

5.2.6 Heuristic 6

Nielsen: Recognition rather than recall.

General: Constructivism/familiarity.

5.2.6.1 IS perspective:

It is easier for end-users to recognise features (screens, sequences, icons, names, words, functions, etc.), instead of having to memorise and remember the features. This implies that users should be familiar with an interface, or be able to recognise familiar screen layouts, task sequences, and meaning – hence, a familiar ontology overall.

5.2.6.2 General life cycle perspective:

According to Akhras and Self, constructivism can be defined as follow:

> "Constructivist theories of learning emphasise an active and autonomous role for learners to construct their own understanding through interacting in an environment in which the knowledge of the domain is not explicitly separated from the context in which it applies." [114]

Constructivism is relevant to both IS and general perspectives. From an end-product and system perspective, constructivism implies that the end-user should be able to learn to execute and understand tasks by interacting with the product. To this purpose, the use of foreign concepts should be limited. If using foreign concepts can't be avoided, familiar concepts relating to the foreign concepts should be available to guide users and stakeholders.

In terms of the system life cycle, all developers, manufacturers, operators and maintenance users should be able to familiarise with technology, processes and procedures, documentation standards and other system elements. Using documentation (typically), stakeholders and users should be referred to familiar concepts when defining, managing and implementing requirements, for example. Model-based systems engineering [115] is an excellent example of using familiar concepts in the form of design models as opposed to using document-based systems engineering.

5.2.7 Heuristic 7

Nielsen: Flexibility and efficiency of use.

General: Balance between agility and efficacy.

5.2.7.1 IS perspective:

From an end-product perspective, decision support, within predetermined constraints and with the option to exercise options, should be available to users and stakeholders. Decision support should assist the end-user in such a way that excessive effort is eliminated as much as possible, but still allowing flexibility. This includes limiting the number of selections, choices, etc. but still allowing sufficient options to support flexibility – hence a balance between minimalism and flexibility. Decision support can be improved with automation and available information. Multiple options to achieving a goal or executing a task should also be allowed in the product.

5.2.7.2 General life cycle perspective:

A combination of decision freedom, options, and limitations, should be available to all users defining requirements and specifications, as well as designers of the product or system.

Clients are prone to change their requirements. Thus the resources and requirements management personnel and systems should be agile. To help control this change in requirements, giving a client uncontrolled freedom can (for example) be changed into giving a client realistic options.

5.2.8 Heuristic 8

Nielsen: Aesthetic and minimalist design.

General: Complexity reduction (Occam's razor).

Occam's razor states that "the simplest solution is almost always the best" [116]. In terms of system design, it calls for design to be done as simplistically as possible. This will improve building, operating, and maintaining the system [117].

5.2.8.1 IS perspective:

Simplicity and aesthetics are important requirements for effective user interfaces. A user interface containing excessive complexity, information, foreign concepts, or confusing content will confuse and discourage the end-users from using the product.

Aesthetics include the structure, format and location of information, use of colours, and other important aspects. Improved aesthetics will improve acceptance by the client and end-users. Hence, the product or system itself must be designed for aesthetic acceptance and should be minimalistically designed.

5.2.8.2 General life cycle perspective:

Limiting the number of system elements helps the end-user, manufacturing team, and the development team with design and task execution. It also improves the product, and system efficacy and reliability as fewer components lead to improved reliability.

Simplicity should be maximised and the complexity limited for the representation of a system. To do this, the system hierarchy with layers of complexity provides part of the solution. By using layered thinking, complexity is reduced in each layer and users are exposed only to layers relevant to them. For example, detail designers are less concerned with operational

elements, and systems engineers are less concerned with detail encountered at the lowest level.

5.2.9 Heuristic 9

Nielsen: Help users recognise, diagnose, and recover from errors.

General:

- Validation, verification,
- Configuration management.

5.2.9.1 IS perspective:

When an error occurs, the product must assist and guide a user with appropriate feedback and guidance from the user interfaces. The guidance from documents and field support should support the user in recognising, diagnosing, and resolving the error.

In the error recovery process, recovery options should be available for users and stakeholders, with diagnosis features assisting the users and stakeholders. The principle of recovering from an error comprises the detection of an error (different from preventing an error), followed by a process of error recovery.

5.2.9.2 General life cycle perspective:

a. Validation and verification play a key role in identifying (detecting) design errors during the development process. Quality control via validation and verification identifies design errors and allows the development team to resolve the errors.
b. Configuration- and change management is the process of error recovery and needs to be specific and allow to change the system configuration. This process is a thus a formal process of error recovery, with most errors ascribed to humans at system operational level. That is incorrect requirements, design errors, test, production and maintenance errors.

5.2.10 Heuristic 10

Nielsen: Help and documentation.

General:

- Support,
- Communication.

5.2.10.1 IS perspective:

Users and stakeholders should be able to access help from the product or program's user interface, as well as from documentation such as user and training manuals. Documentation is used to communicate product functionality, features and status – hence, the need for effective communication is key to usability.

5.2.10.2 General life cycle perspective:

a. Complex systems require specialist assistance from different role players, specifically specialists, during the development phase. This will prevent design errors, improve the development team and end-users' experience, and ultimately reduce development time and risk.
b. Communication forms part of support in various forms, including documentation. From a system perspective, communication is used in project management (according to the project communication plan), systems engineering (requirements, models, designs, etc.) and in general.

From the above analysis, it is clear that Nielsen's heuristics, as applied to software systems' usability, can be generalised in a sensible way to address usability in a system life cycle, and not only the usability of the end-product. It is clear from the analysis that systems engineering, as such, provides processes, methods and tools to support usability. In addition, it is interesting to see that additional usability characteristics, such as contextualisation, complexity reduction and the balance between agility and efficacy also play roles in system life cycle usability.

5.3 Addressing life cycle usability issues with general life cycle usability

From the generalisation of Nielsen's usability of software products to usability from a system life cycle perspective, it is instructive to see how general life cycle usability can address the issues identified in the literature study.

Life cycle usability issues and challenges were identified from literature and discussed in Section 4.1.4. Each issue is listed (as from literature), and the applicable general heuristic applied to assess if the general heuristic will address that particular issue.

5.3.1 General life cycle usability issues

The issues discussed in this section are issues that are common to all life cycle phases.

Information: a) Change management and version control issues:

- Requirements management (3ii): Manage the changes made in the requirements, specifications, manufacturing data packs, etc.
- Validation and verification (9i): Validating and verifying information improve the quality of information and identify requirements management issues.
- Configuration management (9ii): Efficient requirements, specifications, and change management allow freedom for change, within limitations, and manages the effects of the changes.

b) Quality control and standardisation of information, documentation, and designs:

- System life cycle quality (3i): Manage the quality of requirements, documentation, and designs to prevent design, manufacturing, operations, and maintenance errors.
- Consistency and standards (4): Ensure consistency and standards are met for documentation and system designs.
- Validation and verification (9i): Validating and verifying that requirements are met leads to good product quality.

c) Lacking information access, visibility, and representation:

- System life cycle representation and status (1): Efficient system representations and information formats improve the visibility of information, and convey information more efficiently.
- Contextualisation and accuracy of real-world system representation (2): Accurate representations of the real-world system improves the accuracy, visualisation and efficiency of information.
- Communication (10ii): Efficient communication ensures proper information flow and access.

d) Information security vulnerabilities:

- Technical risk management (5): Vulnerabilities in the information system should be identified and managed, even prevented if possible.

Transparency and traceability: Inadequate transparency and traceability of information:

- System life cycle representation and status (1): An accurate system representation provides clear traceability of information.
- System life cycle quality and requirements management (3): Managing the quality of information and reasoning behind decisions will improve traceability and transparency.
- Validation and verification (9i): Testing the validity of information provides a way to trace the origins and quality of information.
- Communication (10ii): Efficient communication will improve the flow and quality of information.

Decision support: Lacking decision support:

- System life cycle representation and status (1): Accurate and efficient system representations and status information provide system information for decisions.
- Requirements management (3ii): Managing requirements, specifications, and information provide information for decisions
- Support and communication (10): Adequate supporting information from stakeholders, including operations and maintenance support, and efficient communication provide information for decisions.

Communication: a) Ontology, terminology, and language discrepancies:

- Contextualisation and accuracy of real-world system representation (2): Relevant ontologies should be used for each stakeholder and user. The systems engineer is crucial for this purpose.
- Constructivism/familiarity (6): Familiar terminology should be used to ensure that each stakeholder would understand the information.
- Communication (10ii): Users and stakeholders should consider differences in ontologies when communicating with other users and stakeholders. If language is a problem, translators may be necessary to avoid misunderstandings.

Technical risk management: Inadequate technical risk management:

- System life cycle quality and requirements management (3): Managing the requirements and its quality will improve risk identification and management.
- Technical risk management (5): Following standard technical risk management methods like FMEA and FMECA will improve the problem.
- Validation, verification and configuration management (9): Validating and verifying information, and proper change management is crucial for ensuring that risks are managed adequately.

Resources: A lack of resources and tools, inadequate tool functions, and tool usage guidance:

- System life cycle representation and status (1): Adequate system representations indicate the necessary resources and improve resource management.
- Requirements management (3ii): Requirements management provides information for resource identification and management strategies. The requirements of tools are also communicated.
- Validation and verification (9i): Validating and verifying that resources are present and managed efficiently.
- Support (10i): Proper training and documented guidance should be present for efficient tool usage.

Collaboration: Inadequate interdepartmental collaboration:

- Requirements management (3ii): All departments' requirements should be attended to, and efficient RM will inform different stakeholders what the design parameters are.
- Support (10i): Departments should provide support to other departments to ensure that all designs and operations between departments function together.
- Communication (10ii): Efficient communication will create an environment which improves collaboration.

A specific issue not directly (but indirectly) addressed by general usability heuristics, is that of resistance to change. However, it may be argued that the provision of contextualisation and a holistic view as obtained from systems engineering, configuration management, and effective communication will assist in reducing resistance to change (with the addition of good management, which is a prerequisite for effective engineering management, which falls outside the scope of this research).

5.3.2 Requirements Analysis phase

The following issues were identified from literature as being specific to the Requirements Analysis phase.

Information: a) Primarily use of text-based formatting are insufficient for complex systems:

- System life cycle representation and status (1): Making use of additional formats like system diagrams improves the representation of system information.
- Complexity reduction (8): System hierarchies and functional analysis reduce the system complexity to a level that is easy to manage.
- Communication (10ii): System functional analysis and system diagrams are improved ways of communicating and representing information.

b) Lacking information context (background, problem statement, etc.):

- Contextualisation and accuracy of real-world system representation (2): Proper background information and problem statements provide additional context to improve the systems requirements.
- System life cycle quality and requirements management (3): Managing system requirements and their quality efficiently improve the flow of accurate information.

c) Inadequate requirements, including invalid premises and assumptions:

- System life cycle quality and requirements management (3): Managing requirements and their quality efficiently reduce the likelihood of incorrect information.
- Validation and verification (9i): Information, including system requirements, should be validated and verified by relevant stakeholders, including clients, to prevent incorrect information.
- Communication (10ii): Efficient communication improves the flow of accurate information and eliminates invalid premises.

Communication: a) Low client involvement:

- Requirements management (3ii): Managing requirements will improve communication and motivate clients and managers to partake increasingly and with efficiently.
- Balance between agility and efficacy (7): The client's voice is included in the requirements, through appropriate options, rather than uncontrolled freedom.
- Communication (10ii): An efficient line of communication should be present between the client and other stakeholders to ensure that the voice of the client is included.

b) Inadequate follow-up:

- Communication (10ii): Efficient communication includes supervisors ensuring that decisions and changes are properly understood and implemented.

Requirements: Lack of non-functional requirements, including usability:

- Contextualisation and accuracy of real-world system representation (2): Having an accurate and real-world representation in the requirements would lead the writers and moderators of the user requirement specification (URS) to improve DFU.
- Requirements management (3ii): Requirements should be managed to include non-functional requirements in design functions.
- Consistency and standards (4): Using standard and consistent usability designing methods add to DFU.
- Technical risk management (5): Managing technical risks will identify and manage possible usability risks in the URS.

A specific issue not directly (but indirectly) addressed by general usability heuristics, is that of sign-off issues. This includes supervisors not reading documents thoroughly before accepting and signing the documents. However, it may be argued that the provision of validation, verification and a holistic view as obtained from systems engineering and effective communication will assist in reducing sign-off issues (with the addition of good management, which is a prerequisite for effective engineering management, which falls outside the scope of this research).

5.3.3 Design, Implementation, and Testing phase

The following issues were identified from literature as being specific to the Design, Implementation and Testing phase.

Information: a) Inadequate requirements, WBS, progress reports, and task allocations:

- System life cycle representation and status (1): Accurate system representations lead to accurate system requirements and WBS.
- Requirements management (3ii): Managing requirements effectively prevents inadequate system requirements.
- Validation and verification (9i): Requirements, WBS, progress reports, and task allocations should be validated and verified to be accurate and up to date.

b) Insufficient test data, including input data for tests:

- System life cycle representation and status (1): System diagrams improve the quality and accuracy of test input data, which will provide more accurate test output results and reduce design errors.
- Validation and verification (9i): Test input and output data should be validated and verified by relevant stakeholders and against the system requirements.

c) Insufficient information validation and verification by stakeholders, specialists, and clients:

- Support and communication (10): Stakeholders, specialists, and clients should provide sufficient support in the validation and verification of information, calling for adequate communication between the development team and other stakeholders.

Communication: Low client involvement in system designs:

- Contextualisation and accuracy of real-world system representation (2): Adequate information contextualisation makes it easier for clients to understand the information, and thus improving the client experience.
- Balance between agility and efficacy (7): Valid options rather than uncontrolled freedom should be provided to the client for efficient client contributions.
- Communication (10ii): Efficient communication to and from the client provides an efficient flow of information.

Collaboration: a) Issues integrating different design sections:

- System life cycle representation and status (1): System representations show how the different design sections function and should be integrated.
- System life cycle quality and requirements management (3): The system requirements, especially for the system interfaces, should be managed well and accurate to prevent integration issues.
- Validation and verification (9i): The design sections should be tested individually and integrated to ensure functionality.

b) Insufficient use of specialist consultation:

- Support (10i): Specialist support should be included where needed to prevent design errors.

5.3.4 Product Manufacturing phase

The following issues were identified from literature as being specific to the Product Manufacturing phase.

Planning: a) Inadequate scheduling and resource management:

- System life cycle representation and status (1): Accurate system diagrams, including functional analysis, shows what resources are needed when.
- Requirements management (3ii): The system requirements should inform management teams which resources to employ when or enable management teams to derive this information accurately.
- Validation and verification (9i): The resource lists and schedules should be validated and verified to be accurate and complete.

b) Unplanned issues like labour strikes and transportation issues:

- Technical risk management (5): The technical risk analysis done in the Requirements Analysis phase should include plans and measures for issues like these, at least as inclusive as possible. Unexpected issues should be managed as effectively possible.

Manufacturing requirements: a) Insufficient manufacturing requirements and designing for manufacturing:

- System life cycle representation and status (1): System representations should include accurate manufacturing representations to improve manufacturing requirements and design considerations.
- Requirements management (3ii): The system requirements should be managed to include manufacturing considerations.
- Support (10i): Manufacturing specialists should be consulted for improved manufacturing requirements and design considerations.

b) Inadequate control and monitoring, including “real-time” and automation control:

- System life cycle representation and status (1): Appropriate system representations improve system control, and status information provides monitoring information, including “real-time” status updates.

c) A lack of flexibility, robustness and reliability in production systems, including human-machine-robot systems:

- System life cycle representation and status (1): Efficient system representations provide information for system engineers and development teams to design for robustness and error management.
- Technical risk management (5): Proper risk identification and management improves robustness and error management.

Information: Inadequate integration and synchronisation of manufacturing processes and information:

- System life cycle representation and status (1): Accurate system representations indicate the manufacturing processes timeline and the status of those processes.
- Validation and verification (9i): The manufacturing information should be validated and verified to be accurate and complete.
- Communication (10ii): Proper documentation and verbal communication improve this issue.

Environmental footprint: Too large environmental footprint during manufacturing, including resource consumption:

- Contextualisation and accuracy of real-world system representation (2): Accurate real-world system representations indicate places in the manufacturing phase where the environmental footprint can be reduced.
- Requirements management (3ii): The system requirements should include environmental principles and reduced resource consumption.
- Balance between agility and efficacy (7): An efficient balance between costs, required resources, and environmental impact should be maintained.

5.3.5 Operations and Maintenance phase

The following issues were identified from literature as being specific to the Operations and Maintenance phase.

Information: Inadequate and inefficient data quality, collection, and processing:

- Requirements management (3ii): The system requirements are managed to include data quality, collection, and processing requirements.
- Validation and verification (9i): The quality of data, and the collection and processing functionality, should be validated and verified to be accurate and up to standard.

Communication: Insufficient guidance for installers, operators, field support, technicians, specialists, and end-users:

- System life cycle representation and status (1): Validating and verifying the system status is essential for efficient product operations.
- Contextualisation and accuracy of real-world system representation (2): Sufficient context and appropriate ontologies in support systems help users with system operations, error management, and maintenance.
- Constructivism/familiarity (6): Constructivism in user interfaces, and user and technical manuals help users to operate and maintain the system efficiently.
- Configuration management (9ii): Users should be guided to identify, diagnose, and resolve errors.
- Support (10i): Support, communication, and documentation should be available, understandable, and up to standard.

Flexibility: A lack of adaptivity in product maintenance strategies:

- Balance between agility and efficacy (7): Flexibility, the efficacy of use, and agility allow for changes in maintenance strategies.

Maintenance: a) Numerous factors in maintenance complicates maintenance strategies:

- System life cycle representation and status (1): System representations indicate maintenance factors to be included in maintenance strategies.
- Requirements management (3ii): System requirements should include and manage maintenance factors, where possible.
- Complexity reduction (8): Reducing the system complexity simplifies maintenance factors and thus simplifies maintenance strategies.

b) Inadequate maintenance is performed:

- Validation and verification (9i): The product maintenance quality should be validated and verified to prevent insufficient maintenance.
- Configuration management (9ii): End-users and technical personnel should be guided to perform effective maintenance with effective change management.
- Support (10i): End-users should have access to documented support, technicians and field support for effective maintenance.

c) Insufficient operations and maintenance (O&M) requirements, as well as designing for maintainability:

- System life cycle representation and status (1): System representations should include accurate O&M representations to improve O&M requirements and design considerations.
- Requirements management (3ii): The system requirements should be managed to include O&M considerations.

d) Insufficient reliability, user control, and automation:

- System life cycle representation and status (1): Efficient system representations provide information for system engineers and the development team to design for robustness, control, and error management.
- Technical risk management (5): Proper risk identification and management improves robustness and error management.
- Balance between agility and efficacy (7): Systems should provide appropriate options, within predefined limitations to allow the end-user to have adequate control.

Diagnostics: a) Inadequate system monitoring and diagnosis:

- System life cycle representation and status (1): The current system status should be available and visible to the end-users and maintenance personnel.
- Validation and verification (9i): The system status should be validated and verified, and if system errors are present, the error diagnosis should also be validated and verified.

b) Failure analysis is a time-consuming process and requires specialist consultation:

- Configuration management (9ii): The user interfaces, user manuals, and technical manuals should guide the user to perform efficient failure analysis. Automation reduces diagnosis and maintenance time and efforts for the user.
- Support (10i): End-users should have access to documented support, technicians and field support for effective maintenance.

Environmental impact: Too large environmental footprint for operations and maintenance, including resource consumption:

- Contextualisation and accuracy of real-world system representation (2): Accurate real-world system representations indicate places in the operations and maintenance phase where the environmental footprint can be reduced.
- Requirements management (3ii): The system requirements should include environmental principles and reduced resource consumption.
- Balance between agility and efficacy (7): An efficient balance between costs, required resources, and environmental impact should be maintained.

5.3.6 Product Retirement phase

The following issues were identified from literature as being specific to the Product Retirement phase.

Planning: a) Insufficient retirement requirements and designing for retirement:

- System life cycle representation and status (1): System representations should include accurate system retirement representations to improve system retirement requirements and design considerations.
- Requirements management (3ii): The system requirements should be managed to include system retirement considerations.
- Support (10i): Recycling and disposal specialists should be consulted for improved system retirement requirements and design considerations.

Disassembly: Difficulties disassembling products due to complexity, a lack of guidance, or insufficient disassembly designs:

- Contextualisation and accuracy of real-world system representation (2): Accurate real-world system representations improve designing for efficient disassembly. Appropriate ontologies help users to understand the disassembly process.
- Complexity reduction (8): Reducing system complexity simplifies the disassembly process.
- Support (10i): Technical support and documentation regarding disassembly should be adequate and up to standard.

Environmental footprint: a) Too large retirement environmental footprint, including recycling, disposal, and reuse of components:

- Contextualisation and accuracy of real-world system representation (2): Accurate real-world system representations indicate places in system retirement where the environmental footprint can be reduced.
- Requirements management (3ii): The system requirements should include environmental principles and reduced resource consumption, including recycling, disposal, and reuse principles.

b) Difficulties in converting environmental principles into design principles:

- Requirements management (3ii): Manage requirements to allow for environmentally green principles. Environmental principles are given acceptance criteria to convert them into functional requirements.
- Consistency and standards (4): Make use of consistency and standards regarding environmentally green principles, including environmental impact assessments.

Re-engineering: Lacking access to system information and usage data for re-engineering purposes:

- Support and communication (10): Sufficient support, documentation, and communication are essential to resolve this issue and help the new development team to improve the original product.

5.4 Action Design Research reflection on general life cycle heuristics

From a research perspective, it is necessary to reflect on the work done up to and including the sections above. The main research challenge is the lack of a full life cycle view of usability. To this end, the main solution (and artefact) of this research is a life cycle usability framework.

The ADR method allows for the development of an artefact while conducting research. In this case, research was done on general issues over a system life cycle that can be linked to usability, as well as existing methods for ensuring usability in products and systems.

Literature showed that several issues exist, as discussed above. It also showed that heuristics are mostly limited to products' interfaces, and limited research was available on general life cycle heuristics. Therefore, the most comprehensive set of heuristics (Nielsen) was used as a reference for the creation and validation of a generalised set of life cycle heuristics, on which an IEEE article was published for peer review [118].

The general life cycle issues were then recalled in this chapter, and the general life cycle heuristics were used to show that they are applicable to the issues. It is acknowledged that

the list of issues is not comprehensive and that there may be additional heuristics required (specifically at managerial level). However, in the context of ADR and DSR, the first iteration of research was conducted in this research to show that an artefact could be designed (derived) by using a valid usability heuristic baseline. The design process follows the ADR process, as outlined in Chapter 2. The end result is a general life cycle heuristic baseline that may be used to address issues as identified from literature.

5.5 CASE STUDY: CENTRE PIVOT IRRIGATION SYSTEM

5.5.1 Purpose

A case study is used to enhance the general life cycle heuristics framework quality, as well as to apply the (well-known) Nielsen heuristic framework to the actual product design. In the ADR research setting, the participation of the researcher was thus to observe and record the roles and functions of different participants as a passive observer.

The development of a centre pivot irrigation system (CPIS) is used in this case study. This is only done up to the point where the system is being field-tested because of time limitation. Afterwards, the life cycle heuristics are updated against the observed data.

The usability framework defined in Section 5.1 to 5.4 was applied to the development of the CPIS. Due to confidentiality factors, the Engineering Company (EC) and Pivot Vendor (PV) will not be disclosed, but instead referred to as companies EC and PV. The case study shows how the usability framework may be implemented to an IoT system and the value of usability from a full life cycle perspective.

5.5.2 Case study details

The case study was conducted from February 2018 until November 2019 in Potchefstroom in conjunction with a local development company. The researcher played the role of an observer, although the researcher was required to draw up a user manual as part of the arrangement. This was an ideal environment for ADR as the researcher participated (action) while conducting research on the development process, which includes all design phases. The product was successfully developed as described below, was accepted by the integrator and the distributor, and is currently commercially successful and in production.

5.5.3 System analysis

A functional analysis of the CPIS was done to apply the usability framework efficiently. This includes operational level functional flows and system operational architecture.

The system environments of a general IoT system are shown in Figure 23. Figure 32 shows the system environment for the case study. In each environment, the above description is the environment for a general system. The bottom description is the CPIS specific environment. The engineering company (development environment), the client (pivot vendor/farmer), and the manufacturing companies are needed to develop the CPIS (product) that will be used on a farm (product environment). Usability needs to be present in all environments.

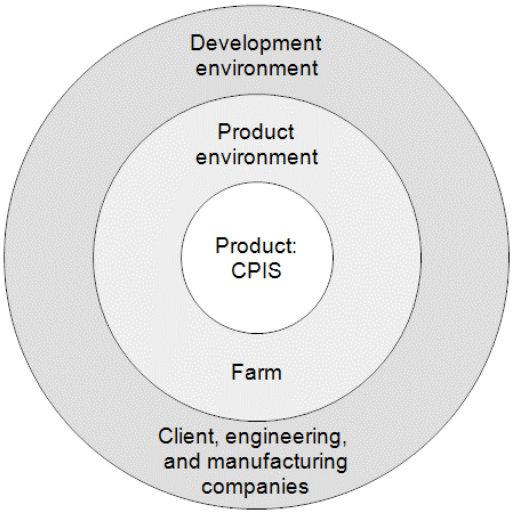

Figure 32: Centre pivot system environment

5.5.3.1 CPIS architecture

Figure 33 shows the operational architecture of the CPIS system. The diagram provides context for the case study as the operational target model for the CPIS is the end goal of the development project. The development team around the product (as shown in Figure 32 above) is to develop the end product. Life cycle heuristics will be verified against the development team efforts to provide the end-product, while Nielsen's heuristics will be verified against the product itself.

To avoid confusion, the observations will be described, after which both the life cycle and Nielsen's heuristics will be discussed in the form of reflection. Business considerations (cost, legal, and other commercial aspects) were not taken into account, and the focus remained only on technical aspects of the development.

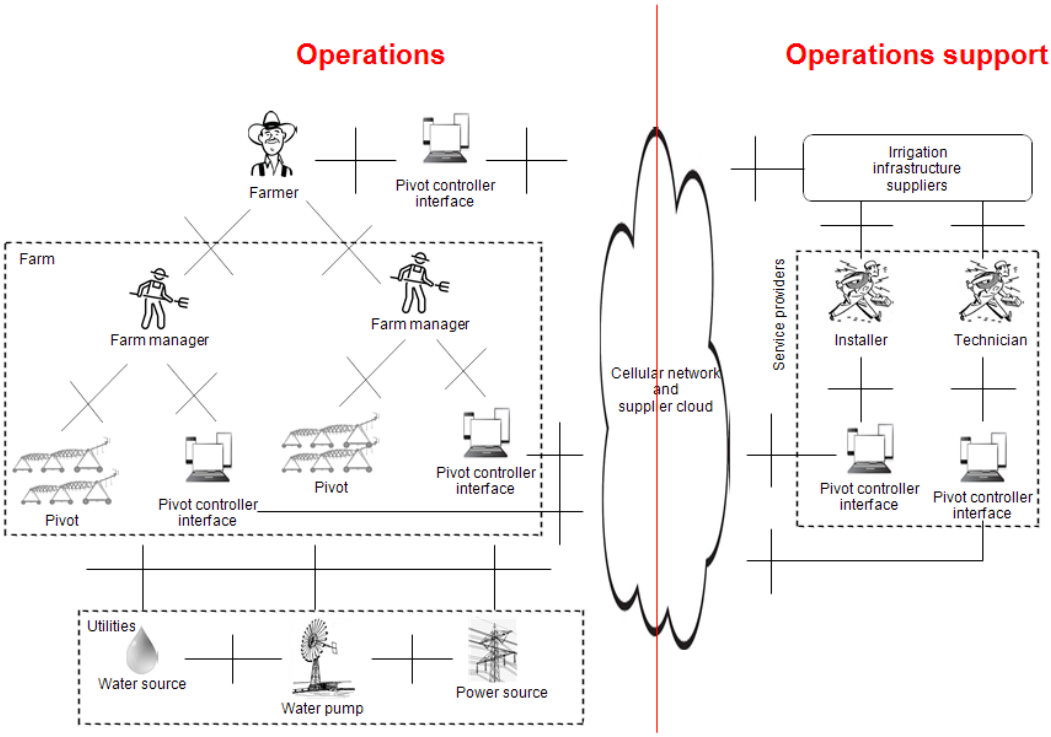

Figure 33: Pivot controller high-level operational architecture

5.5.3.2 Requirements Analysis phase

Overview: Development of the CPIS was initiated by the client as a need existed for reliable CPIS hardware that is interconnected, hence the need for an IoT-enabled system. Initial meetings immediately focused on operations and used functional architectures familiar to the client to represent the CPIS and to elicit requirements. Following workshops included more personnel that focused on preliminary and detail design.

Resources:

In terms of roles and responsibilities, the following functional roles were present:

- Systems engineer: Providing context and elicitation of high-level requirements, limiting technical risk;
- Product development (principal) engineer: Ensuring high-level functional requirements were clearly defined and managing the project;
- Operational domain specialist (mechanical/flow design engineer in this case): Irrigation design, installation and maintenance specialist to ensure an accurate representation of operations data;
- Hardware design engineer: An electronics engineer focusing on rapid prototype development of all hardware elements of IoT systems;

Activities:

The following activities were conducted in order to elicit and define requirements for the CPIS:

- **Initiation meeting:** All role players were present in the initial meetings to discuss both business and technical opportunities and challenges. An operational architecture was drawn up, and the CPIS ontology was shared by the Operational domain specialist. It was clear that the existing equipment was outdated and not connected to a central management system, which is a challenge in the rural areas of South Africa. This defined the design challenge, namely to develop (from a full life cycle perspective) a CPIS IoT-enabled solution;
- **Workshops:** Requirements analysis workshops were conducted where the Product development engineer, Operational domain specialist, and Hardware design engineer were present. Initially, the IoT engineering specialists did not have a full understanding of the agricultural ontology, specifically related to CPIS. The lack of design documentation (development requirements, drawings, test reports, design reports) forced the engineering specialists to conduct a functional analysis on the existing system and to document the system functional analyses (flows and architectures). In addition, development requirements were defined for IoT hardware and software. The network and cloud solutions had existed at the time and were used as part of the overall solution.

Reflection on life cycle heuristics:

From a life cycle perspective, the following was observed when considering how life cycle heuristics were applied in the CPIS project:

- **Heuristic 1 - System life cycle representation and status:** At the onset, the new system was developed using project management and systems engineering principles to show development status. The existing system was a combination of in-house hardware (the CPIS mechanical structure) and an outsourced electronic solution that was limited in terms of functionality. The revision history and related information was not available to show system configuration status;
- **Heuristic 2 - Contextualisation and accuracy of system representation:** The existing system had limited documentation, mostly in the form of manufacturing and user manuals. No system documentation (integrated/holistic) existed, which was mitigated with a new operational analysis being documented based on inputs from the Operational domain specialist and information obtained from the analysed system;

- **Heuristic 3 - System life cycle quality and requirements management:** The voice of the customer is, in this case, the CPIS client whose inputs were used to control the form, fit and function of the end-product. The client was clear on the shortfalls of existing products, but the product definition was done entirely by the development team – this was done by using a requirements management process (Model-based systems engineering);
- **Heuristic 4 – Consistency and standards:** This is a very important life cycle heuristic that was applied in this project using well-defined (consistent) methods for system analysis, documentation, and model representation. This allowed the client to become familiar with the "way of work" and the interfaces (protocols) used between client and developers. The most important contribution towards consistency came from systems engineering methods and tools;
- **Heuristic 5 – Technical risk management (error prevention):** Technical (and associated project) risk management methods were used to identify potential areas of failure, their likelihood and impact. Mitigations then prevented these failures (or errors). This was not done formally in the CPIS research project, although the principles were applied (mostly due to time pressure and urgency). Risks were mitigated by developing a prototype as quickly as possible in order to start field testing and optimisation. Failures in the CPIS project would imply mostly failures of the development resources involved in this phase, which was mitigated by using experienced engineers and by following known, quality-controlled processes;
- **Heuristic 6 – Constructivism/familiarity:** During meetings and workshops, care was specifically taken by the development resources to understand and learn the operational ontology. This was systematically done using systems that existed at the time as references, and by iteratively enhancing operational models from prior models (constructivism). Rapid prototyping was used to demonstrate concepts from which the operational ontology was enhanced (in other words, an iterative and incremental design approach [119] was followed with requirements analysis and implementation being done iteratively). The client was familiar with the development environment, so limited constructivism was evident in this regard;
- **Heuristic 7 – Balance between agility and efficacy:** Initially, the project was under severe time pressure, but the initial phase of Requirements Analysis was not bypassed despite the limited time. However, a model-based systems engineering (MBSE) approach allowed the team to define a concept operating model and associated equipment requirements. A balance was found between agility and efficacy to allow

requirements elicitation to take place, at the cost of initial formalities, such as a documented, signed off User Requirements Statement;

- **Heuristic 8 – Complexity reduction (Occam's razor):** An interesting result followed from the Requirements Analysis due to the initial lack of documentation, namely that the functional analysis that had to be (re)conducted. It allowed the developers to form a simplified abstraction of the system. This, in turn, allowed the analysts to identify opportunities for improvement by using new technology, and effectively leap-frogging existing systems as there were limited electronic technology legacy constraints. Thus, by being forced to effectively "stand back" from the system allowed the analysts to remove complex, outdated technology from the abstraction effectively;
- **Heuristic 9 – Validation, verification, and configuration management:** The ability to detect an error in the requirements phase is critical as errors will propagate to following phases. Errors in the CPIS were detected by interacting with the Operational domain specialist and Design specialists (engineers) to ensure requirements were both valid (true to the actual problem) at the onset. Configuration management was not formally followed in the CPIS project (in this phase, at least) but changes were controlled centrally by the Systems engineer. The client trusted the development team to derive and verify requirements for equipment;
- **Heuristic 10 – Support and communication:** The client provided support for Requirements Analyses in the form of engineers experienced in the Operational domain (they were not part of the team, as such). Communication was managed by the Principal engineer who managed both technical and project management communication. MBSE provided a framework for communication, and the Systems engineering function was central to forming an interface between client and developers. That is, the systems engineering function was used to translate between the operational ontology and the development engineering ontology. The systems engineering model was observed to be central to the communication effort because it was used as a reference in almost all communication, as opposed to a document-based approach.

Summary: From the above analysis, it is clear that a number of improvements/additions to the life cycle usability framework are possible:

1. MBSE is a good alternative to the document-based systems often used during development. The model itself is central to communication of concepts, creation of context, and management of change, amongst other advantages. Therefore, the MBSE method of system representation support usability;

2. The role of the Systems Engineer should not be underestimated as she/he fulfils a critical interfacing function, amongst the usual systems engineering functions. This, in itself, is a life cycle usability factor as virtually all role players (in the CPIS case study) interact with the Systems Engineer;
3. Rapid prototyping is most helpful in a constructivistic way as the development team, and client team all align perspectives based on real-world experiences from the actual prototype. Product characteristics (to match requirements), validation and verification, and technical risk management are all addressed by a prototype, which is thus highly usable in the CPIS context;
4. Reworking a system/product due to limited documentation may be advantageous as working from a new slate allows legacy issues to be addressed, and allows complexity reduction (in this case) due to advances in technological development. This indicates the possibility to enhance (re-engineer) old or outdated systems. In terms of usability in a full life cycle, the system's complexity may be reduced by regular reviews and upgrades, which thus introduces a usability improvement.

5.5.3.3 Design, Implementation, and Testing phase

Overview: This phase was characterised by rapid prototyping using existing software and hardware modules. From a model-based perspective, the focus was on realising a prototype and in the process deriving the required product characteristics. A demonstration model was rapidly developed with usability (resulting from connectivity) as the main focus.

Resources:

In terms of roles and responsibilities, the following functional roles were present:

- Senior systems engineer: Executing all systems engineering functions [6];
- Senior product development (principal) engineer: Ensuring product meets requirements, production planning, verification testing, information design;
- Senior operational domain specialist (mechanical/flow design engineer): Low-level requirements validation;
- Senior CPIS equipment manufacturer (technician): Manufacturing requirements and procedures for consideration in the production process;
- Senior IT engineer (computer engineer): Cloud services development providing infrastructure definition and aligning client requirements with available cloud capabilities;

- Senior network/communications engineer (electronic engineer): Network communication back-end implementation;
- Senior hardware design engineer (electronic engineer): Simulations and modelling, electronic circuit design, hardware interface design;
- Junior product design engineer (electronic engineer): Technical assistance with electronic design, user interface design;
- Junior product design engineer (computer engineer): Web application interface design;

Activities:

The following activities were conducted in order to design, implement and test the CPIS:

- **Hardware design:** Dedicated hardware development was done using existing hardware modules and implementing new functions from the Requirements Analysis. The Principal engineer and Hardware electronic engineer were mostly responsible, with assistance from less experience electronic engineers to realise printed circuit boards (PCBs) and hardware applicable specifically to the CPIS product. From a usability perspective, the hardware user interface was designed using Nielsen's principles. The hardware prototype was used as a physical realisation of the systems engineering model. It was, from a usability perspective, designed to have usable human-machine interfaces and to be inherently safe, maintainable, and robust. Robustness and built-in diagnostics support usability as these allow end-users (farmers, foremen, maintenance personnel) to operate and diagnose products in the field remotely. Design for functional capability, manufacturability, reliability and usability was done;
- **Software design:** Both cloud software and firmware development were done, mostly using existing infrastructure. Cloud software used existing message switching, storage and processing, amongst other functions. User interfaces were applied (in the case of existing cloud software) and developed for application-specific use. A web interface was created to allow remote access from fixed and mobile devices. Software human-machine interfaces were designed using Nielsen's principles. It is instructive to understand that an ineffective interface can lead to highly undesirable consequences as CPIS equipment are directly controlled using software interfaces, hence the need for clearly defined interfaces. Design for functional capability and usability was the key focus;
- **Network design:** As a cellular network was used, the GSM standards were applicable. The network relied on existing infrastructure in the form of firmware, cloud software and cellular network provider infrastructure. The protocol between edge devices

(controllers) and cloud was defined at application level only as the lower levels already existed. This protocol includes both operational and maintenance information to be displayed, controls available to end-users based on access levels, as well as system logs available to engineers for operational support. As the human interfaces rely on the protocol, alignment was achieved by ensuring good communication between application, network, cloud and firmware developers. Design for functional capability, reliability (availability), and usability was the main focus;

- **Test equipment and procedure design:** As part of the actual system life cycle design, test equipment and procedures were designed in this phase to ensure product manufacturability was achieved. The use of test equipment not only speeds up manufacturing but also limits human error due to the automated nature of testing. Usability played a role as the human interfaces to test equipment was implemented using standards used in the manufacturing industry.

Reflection on life cycle heuristics:

The following was observed for design, implementation and testing when considering how life cycle heuristics were applied in the CPIS project:

- **Heuristic 1 - System life cycle representation and status:** Design was done using the MBSE approach with extensive modelling and simulation having been done before physical implementation. The use of MBSE implies that the model itself was used to represent the system, followed by the actual implementation. In this rapid prototyping environment, there was limited time for a document-based approach, and the MBSE approach was found to represent the system and its configuration status sufficiently;
- **Heuristic 2 - Contextualisation and accuracy of system representation:** The context of the system, to be reminded, includes the development environment as well as the product under development. The system context is thus less relevant to the product and more to the development environment, which must be monitored by the Principal engineer for progress and quality. Therefore, the key interface that must be usable is between manager and team, as well as between manager and product – this requires the human interfaces provided by a good project manager and systems engineer, respectively;
- **Heuristic 3 - System life cycle quality and requirements management:** Constant interaction with the client addresses the usability requirement of allowing the voice of the customer to be heard. In this case, the “customer” actually includes all relevant stakeholders as there are more beneficiaries to the system than just the client – requirements management ensured all voices/inputs were taken into account. The

requirements included “design for” requirements, which were obtained from the development team’s knowledge base. The fact that all relevant users were able to provide design input, and have visible evidence of this, was a motivating factor;

- **Heuristic 4 – Consistency and standards:** Design and implementation of the CPIS were done using an industry-standard CAD application for hardware design, and standard electronic and electromechanical components and machine-machine interfaces. The design is based on equivalent systems in the market and interfaces in industry-standard ways to the pivot structure, pumps and sensors. GSM standards were implemented, software development standards were followed using GIT as repository and Wrike as project management tool. An iterative and incremental development (IID) approach was followed, and functions were added sequentially. The systematic improvement and feedback provided visual evidence of progress, which was found to be a motivating factor. By following standards, consistency was introduced, and it was possible to achieve results quickly, which is inherently gratifying;
- **Heuristic 5 – Technical risk management (error prevention):** By using a rapid prototyping approach on both hardware and software, it was possible to identify design and implementation errors early in the process. Operational shortfalls were easily identified early in the process as the prototype provided an almost immediate means of validation. Although risks were not managed on a risk register, potential failures were identified and mitigated, and issues were continuously monitored and addressed. Usability of the process was thus provided in that design errors, and other human failures were not allowed to escalate into major failures, which in themselves are inherently stressful and demotivating;
- **Heuristic 6 – Constructivism/familiarity:** Designers were able to use existing hardware and software modules, which resulted in a highly constructivist approach as the designers were both familiar with modules and were able to build upon these. Modularity in design and implementation simplified design and allowed the core system to be reused in a very short time. This then simplified the remainder of the development (mainly interfaces, specialised electronics, and business rules) in a shorter time as the designers were able to spend time on new concepts with which to familiarise. Usability in design and implementation (as a life cycle phase) is thus supported by modularity, which is in turn supportive of both constructivism and familiarity;
- **Heuristic 7 – Balance between agility and efficacy:** Designers were allowed to be flexible in the design process (selecting components, interface modules etc.), but still followed a structured approach guided by design constraints and other requirements.

Hence, creativity was allowed inside the bounds of requirements, which was found to provide a balance between agility and efficacy. Usability was thus provided by allowing inputs into the design but limiting the options by applying requirements (form, fit, function) and design constraints;

- **Heuristic 8 – Complexity reduction (Occam's razor)**: By using a model and rapid prototype, the system development complexity was reduced as a document-based approach was not followed (which is highly complex). By using a layered approach, it was possible to limit complexity at higher layers, and repetitively through lower layers throughout the design process – this reduced complexity for design, implementation and testing. MBSE allowed focus on visualised sub-systems, modules and components and their interfaces (limited number of elements per sub-system) system under development, which, combined with layered thinking reduced the number of components under consideration in each layer;
- **Heuristic 9 – Validation, verification, and configuration management:** Initial validation of requirements (in the Requirements Analysis phase) resulted in fewer iterations in this phase. For this reason, the upstream activities increased usability downstream. Rapid prototyping with an Agile approach (simulations before construction to reduce physical hardware iterations) resulted in shorter verification time periods. Configuration management of software was done using tools such as GIT for version control and for PCB layouts and simulations (using Altium PCB design tools). Because of the limited size of the team, communication was simplified, and changes were managed more easily;
- **Heuristic 10 – Support and communication:** Design and implementation support were provided by component suppliers (field application engineers), software development forums, experienced engineers (as part of the team), as well as by development tools (PCB layout CAD, software IDE's, oscilloscopes and others). In addition, design documents and historic designs (from the body of knowledge) was found to be highly supportive. Communication was critical as alignment between team members was critical in the relatively closed, small development team. This took place using Wrike for project management, regular project meetings, and informal communication managed by the Principal engineer, and between different focus groups (software, hardware, and product).

Summary:

The following key points add to the concept of usability during design:

- The use of MBSE to improve usability is confirmed in this phase as the focus of the design is on the model of the actual product being designed and implemented. The model (and all its meta-data) supports visual forms of communication, layered thinking (complexity reduction), configuration management, constructivism and finally, contextualisation;
- Rapid prototyping is an excellent method to ensure constructivism, support validation and verification due to immediate feedback on fit/form/function and performance, and to ensure the client is drawn into the process to add her/his voice;
- A good Systems Engineer and good Project Manager support communication and collaboration, which makes the design, implementation and verification process usable to all participants. The technical interface between client/operations and specialist developers is formed by the Systems Engineer, while the Project Manager interfaces between development resources on time, cost and quality;
- The iterative and incremental development approach (where functionality is specified initially and added iteratively), supported by rapid prototyping and support tools (such as CAD for modelling), also supports constructivism and verification. A goal-driven approach was followed since all functions were defined at the onset, as opposed to an Agile approach where requirements are typically less defined initially.

5.5.3.4 Product Manufacturing phase

Overview: This phase focuses on usability during production, including aspects relating to individual item manufacturing (hardware and software) and integrated item manufacturing or rather, assembly and integration. Design for manufacturing (in prior phases) is used to ensure usability in this phase.

Resources:

In terms of roles and responsibilities, the following functional roles were present:

- Product engineer (electronic engineer): Transfer of manufacturing data pack to contract manufacturer;
- CPIS equipment manufacturer (team): Manufacturing of switchgear and quality assurance (integration testing);

- Contract manufacturer (team/contractor): Manufacturing of control boards, position encoders, pivot controller user interface;
- Product design engineer (electronic engineer): Test equipment for production testing (bed of nails) and associated test software;
- Test engineer (electronic engineer): Test firmware and software for unit configuration and production testing.

Activities:

The following activities were conducted in order to manufacture the CPIS:

- **Hardware manufacturing:** Design software (Altium) compiles a manufacturing data pack for each PC board. Different modules are compiled into a single data pack, which each module being manufactured on the production line a single item. Single items are tested after which they are programmed (where applicable) and configured. Items are integrated first into sub-assemblies (assemblies of items/modules) and tested, and then transported to a system integrator where final integration and testing are done. The integrated assembly is then packaged and shipped to the pivot original equipment manufacturer (OEM) where the electronic control unit, electrical sub-system, and mechanical sub-system are assembled. Hardware diagrams are included in the box to assist in fault finding and to provide information in later phases.
- **Software programming and configuration:** Programming of the pivot controllers is done at the design house (initially, to ensure the quality of the first production units). It will be transferred to the product manufacturer once fully verified. Part of configuration is to download diagnostic software onto production units. Web-based applications are used to increase usability as end-user mobile device programming is thus not necessary (not an application);
- **Testing:** PCB manufacturers perform bare board testing to ensure quality before components are placed. Production testing follows a systematic process where functions are tested according to a test sheet with a table of hardware circuits and more complex functions to be verified. A full test specification is provided to a trained test technician, including processes, procedures, specific instructions, test tables and conditions, fault finding and diagnostics. A diagnostic feature (software) is used to simplify testing and to automate the test process, where all sensors and other inputs become visible to the tester. The system integrator performs functional testing at a higher level than the electronic board level and includes testing of electrical/electronic/mechanical interfaces. A full functional test is done to ensure all

interfaces and hardware/software elements are tested after integration. Serial numbers are added, and quality control labelling is provided;

- **Packaging and logistics:** Sub-assemblies are packaged with conductive plastic (such as bubble wrap) to prevent failures from ESD (electrostatic discharge) and shocks. Clearly labelled boxes are used in boxes not too heavy and marked as fragile where necessary. Sufficient packaging is provided as transport personnel are known for rough handling of packages. Finally, assembled controllers (in large, heavy boxes) are transported in protective packaging to the pivot original equipment manufacturer (OEM). A clear indication is provided to indicate the nature of the contents to the transporters (also users).

Reflection on life cycle heuristics:

From a life cycle perspective, the following was observed when considering how life cycle heuristics were applied in the CPIS project:

- **Heuristic 1 - System life cycle representation and status:** The product's status from individual components in the factory to a final assembled product must be represented. In order to achieve this, the overall manufacturing process and product breakdown (in the form of a tree) should be used. In this case study, a complete representation was not drawn up, but rather a piecewise representation applicable to each manufacturing environment, namely (i) a manufacturing data pack for the board manufacturer, containing guides with manufacturing and testing specifications, (ii) assembly data pack for the assembly of modules, and (iii) integration data pack for combining the control unit and the mechanical assembly. Measurement of progress (to show production status) was done manually for the whole system, but production status was tracked in each individual production facility (PCB manufacturer, control unit production, and final integration) using dedicated production management software;
- **Heuristic 2 - Contextualisation and accuracy of system representation:** A manufacturing data pack was compiled by the product engineer and development team. The manufacturing data pack provides context and guidance for manufacturing and testing technicians using appropriate ontologies. Hardware diagrams and system models, developed during the MBSE process, provide accurate system references against which the actual system can be measured. The Principal engineer monitores overall manufacturing progress and deviations from reference, while individual production facilities use references in their respective environments (specifications) to measure the accuracy of modules, sub-assemblies, and assemblies. This is

understood to be manufacturing configuration that supports usability (being able to measure against references);

- **Heuristic 3 - System life cycle quality and requirements management:** The voice of the customer is, in this case, all manufacturing stakeholders, including the client. Manufacturing requirements are managed through manufacturing data packs and quality controls in the manufacturing process. Quality assurance was thus achieved in this case study using clear specifications, where quality control was done in each production specialist's environment, with specific tests and inspections at each point where components, modules, sub-assemblies and assemblies change hands. Also, the fact that the system can run diagnostics via the cloud allowed online checks later on in the production and assembly process, which simplified the quality process significantly;
- **Heuristic 4 – Consistency and standards:** By using standard design software and documentation over an array of projects, the manufacturing data packs and test specifications provided consistent and standard principles, documentation, and models in this project. Also, using standard components simplified the procurement process. Testing utilised standard testing processes, procedures, tools, fault finding, and diagnostics. Standard quality control procedures and documents allowed for consistent quality assurance and control. Standard packaging, including conductive plastic, simplified the procurement of packaging materials;
- **Heuristic 5 – Technical risk management (error prevention):** The manufacturing data pack and test specification were clear and accurate to prevent manufacturing and testing errors. Comprehensive production testing was done on components, single items, sub-assemblies, integrated assemblies, and final products to identify and manage manufacturing and design errors throughout the manufacturing process. Test sheets with a table of hardware circuits and more complex functions were used to ensure all required testing was done. The programming of pivot controller units was initially done at the design house for quality control but would be transferred to manufacturers once fully verified. Sufficient packaging, including conductive plastic, was important for preventing transportation damage. Throughout, risk management was done by identifying potential failure points from experience and the existing body of knowledge;
- **Heuristic 6 – Constructivism/familiarity:** The manufacturing data pack represented the system models and manufacturing processes in such a way that it guided stakeholders to familiarise themselves with manufacturing and testing processes, procedures, specific instructions, fault finding, and diagnostics quickly and efficiently.

Standard components and tools are also used that the manufacturing personnel are familiar with;

- **Heuristic 7 – Balance between agility and efficacy:** Manufacturers were allowed to be flexible in the manufacturing process (adjusting for components procurement issues, transportation issues, and other delays), but still followed a structured approach guided by manufacturing time constraints and other manufacturing requirements. Hence, time-delays were allowed inside the bounds of requirements, which was found to provide a balance between agility and efficacy. Cooperation between the design house and the production houses assisted in allowing changes with limited risk;
- **Heuristic 8 – Complexity reduction (Occam's razor):** By using a layered approach, it was possible to limit complexity through the operational and technical layers throughout the manufacturing process – this reduced complexity for manufacturing and production testing. MBSE allowed focus on visualised sub-systems, modules and components and their interfaces (limited number of elements per sub-system), which, combined with layered thinking reduced the number of components under consideration in each sub-assembly and module. Automation and tools such as diagnosis software simplified the manufacturing and production testing processes by minimising interaction between human and technologically complex production resources;
- **Heuristic 9 – Validation, verification, and configuration management:** Components, single items, sub-assemblies, and final assemblies were thoroughly tested, and functionality verified and validated. The testing was done against the product specifications, which were verified in earlier phases. Testing on different assembly levels helped to identify issues and errors early on. Quality control labelling provided verification of functionality for the downstream stages. Hardware diagrams assist test technicians in fault finding. Diagnostics tools were also used to identify and manage manufacturing issues and errors;
- **Heuristic 10 – Support and communication:** A manufacturing specialist and the Principal engineer provided support in the form of specialised manufacturing and management knowledge. Communication was managed by the Principal engineer who managed both technical and project management communication. The manufacturing datapack (and the MBSE models) provided a framework for communication, and the Systems Engineering function was central to forming an interface between manufacturers and developers. That is, the systems engineering function was used to translate between the manufacturing ontology and the development engineering

ontology. Thus, the manufacturing data pack (including test specifications) and reference design models (from MBSE) were found to be highly supportive with respect to communication.

Summary:

The following findings were made from the manufacturing phase:

- In general, manufacturing processes and procedures have been set up to allow for mass manufacturing, using automated processes and equipment. Hence, the challenge of usability during manufacturing is less pronounced than for the initial life cycle phases, such as requirements analysis;
- The manufacturing datapack plays the most important role to control the production process, as is commonly known and confirmed here. Processes, procedures, schematics, standard bills of quantity, assembly drawings, photographs and packaging instructions form part of communication to the production houses. By following the standards provided by the manufacturers (i.e. considering their ontologies), the datapack supports usability in a very significant way;
- The addition of remote access (via the cellular communications network) to the product in the later stages of manufacturing and integration was found to be highly supportive of usability as diagnostics, remote verification, and configuration of units could be done centrally. Fault finding was simplified as expert assistance was available online;
- Again, the MBSE approach with layered thinking allowed each product layer to be considered almost in isolation and directed to the manufacturer on an applicable layer. To elaborate, board manufacturing took place in the detail layer with test procedures developed by a test engineer on the same layer, based on results from simulations also on the detail layer. In the technical layer, sub-assemblies are configured on that layer and provided to a technician at the CPIS integrator and tested according to specifications developed by a test engineer on the same layer. Finally, the operational layer combines sub-assemblies to form a CPIS assembly, tested mostly by means of inspection and configured in operational terms as defined by requirements at operational level.

5.5.3.5 Operations and Maintenance phase

Overview: This phase focuses on usability during Operations and Maintenance, including aspects relating to installation, day-today pivot operation, and system maintenance. Design for operational usability and maintainability (in prior phases) are used to ensure usability in this phase.

Resources:

In terms of roles and responsibilities, the following functional roles were present:

- Operational domain specialist (engineer): Irrigation design, installation and maintenance specialist to ensure effective product use, training support;
- CPIS equipment manufacturer: Product provision and support with complex installations, training support;
- Field technician: Installation and commissioning of systems and field support, repairs of failures, training support;
- Engineering support (team): Adjustments and feature addition from field changes, operations support (third tier), manufacturing support, technical training;
- Operator: Receiving training, operating CPIS in the field, reporting of failures and errors, basic error recovery;
- Farmer: Receiving training, operating CPIS in the field, reporting of failures and errors, advanced error recovery, management of multiple CPISs;
- Analysts: Detection of fault condition patterns, analysis of maintenance and performance data for trends, operational data analysis (performance data).

Activities:

The following activities were conducted in order to operate and maintain the CPIS:

- **CPIS installation:** Pivots are installed on site following a defined process. Pivots also run on a schedule according to defined processes (as was obtained from the requirements analysis) as programmed into units during production. The technical configuration is done on site during and after installation, followed by on-site user acceptance testing. Farmers program their specific operational settings from the control panel or a remote device (mobile device) via the panel or web-application user interface.
- **CPIS operation:** Pivots operate either manually or automatically according to pre-defined schedule, as per end-user requirement. The CPIS provides status information on user interfaces (a user interface on the control unit or a mobile web application) with the most important status indicators clearly visible. The product user interface is critical, where the design will be discussed in a later section below. Failures in the field (pivots getting stuck, component failures, utility failures including water and electricity supply) are reported by means of error messages and indicators. Control messages are sent to the pivot structures via the mobile web application or local interface;

- **CPIS maintenance:** There are three levels of maintenance (tiers), including first level on the site by the farmer, second level by the field technician or equipment manufacturer, and third level when the engineering design team is required. Status information is clearly visible on the local control unit with fixed-function indicators clearly marked on the user interface and more detailed information available from an LCD display. In cases where the communication has not failed, the end-user may access the panel using a mobile device with similar failure indicators and detail available. Informative systems, such as this CPIS control system, provides information that simplifies repair activities, such as knowing beforehand what failures occurred and the location of such failures. For example, knowing which section of a pivot is faulty allows the repair technician (or farmer) to diagnose and repair the fault faster with less effort.
- **System optimisation:** Business intelligence, from analytics, was used to show recurring failures, frequency of failures, and other failure trends. Also, use patterns that lead to sub-optimal operation are easily identified and rectified by using patterns of user behaviour. For example, user patterns that lead to similar failures may be due to lack of training or overcomplicated user interfaces. Advanced functions may be added to automate and simplify operations according to use patterns.

Reflection on life cycle heuristics:

From a life cycle perspective, the following was observed when considering how life cycle heuristics were applied in the CPIS project:

- **Heuristic 1 - System life cycle representation and status:** The user and technical manuals provided accurate system and product representations for users (farmer, field support, and specialised support) to effectively operate and maintain the product. Informative diagrams were included in the manuals and are a powerful tool in aiding text-based information. For the system status on site, the CPIS user interface was confirmed to be the most critical element (as is known). The most important status indicators were designed to be clearly visible on the controller's user interface. These indicators are static in that their functions do not change. More informative error messages and other details appear on an LCD screen as well as on a web application to notify the user of operational status and failures in the field. From an IoT perspective, knowing the status of the cellular network is in itself the most critical factor, and a fully managed network was developed to ensure comprehensive visibility of network status in terms of status indicators such as link quality, network up-time, and others. Remote access to units in the field allowed improved monitoring of on-site CPIS technical and

operational status, as is the purpose of IoT systems. Also, long-term status indicator trends can be recorded (e.g. utility status, operational performance, and failures amongst others) to inform higher layers in the IoT system hierarchy;

- **Heuristic 2 - Contextualisation and accuracy of system representation:** The user and technical manuals were compiled by the product engineer and development team. These manuals provide context and guidance for farmers and field support using the appropriate ontology. Installation instructions, operations diagrams (including setting up irrigation schedules) and other system models developed during the MBSE process provide accurate system representations for context. The CPIS status indicators provide additional context and familiarity for users, describing commonly known aspects such as power supply, water pressure and the like. Accuracy of the system representation includes the ability to communicate effectively via the user interfaces by using appropriate icons (as part of system status indication), and understandable terminology – the accuracy of the status is thus also reflected in the relevance of the information content of the message to the operators, maintenance personnel and engineering teams;
- **Heuristic 3 - System life cycle quality and requirements management:** The voice of the customer includes, in this case, all operations and maintenance stakeholders. The capability to adjust pivot irrigation schedules and processes was included in the controller programming as per system requirements. This means farmers and field support can set up their individual pivot schedules and irrigation requirements on the CPIS. Failure trends and use patterns are identified by using analytics. These patterns are also a form of input to the optimisation process, as obtained from farmers and field support – information from the patterns are used to create advanced functions that may automate and simplify operations in the future (when considered from a full life cycle perspective);
- **Heuristic 4 – Consistency and standards:** Consistency was provided by means of functional definition and interfaces, providing the end-users with similar processes and interfaces across different platforms. These platforms include the actual control unit on site, web applications, and references in user manuals. The use of standard documentation means that the user and technical manuals flow logically (as best practice dictated) and are aligned with the design documents to ensure consistency across the life cycle phases. Standard installation, operation, and maintenance processes and procedures were defined to be followed, including consistency with other CPIS systems in the market (as expected by farmers, operators, maintenance and field personnel);

- **Heuristic 5 – Technical risk management (error prevention):** The user and technical manuals are clear and accurate to prevent pivot operations and maintenance errors. User acceptance testing is done on the CPIS to identify and manage installation errors. The status indicators and pivot information on the LCD screen assist the user in the identification and management of issues. The pivot controller was also designed to prevent tampering by farmworkers, for example, bypassing the pivot override function. Analytics are used to identify failure trends and use patterns to identify issues such as user training issues and overcomplicated user interfaces. Usability of the process was thus provided in that operations and maintenance errors, and other human failures were not allowed to escalate into major failures, which in themselves are inherently stressful and demotivating;
- **Heuristic 6 – Constructivism/familiarity:** As with consistency, the farmers and operators are used to a specific ontology and environment. The CPIS in operation was designed to present a familiar interface, and the farmer can adjust parameters and configure the solution to his/her needs – hence a constructivistic approach when taking part in the setup. The user and technical manuals represent the operation and maintenance processes in such a way that it guides users to familiarise themselves with operations and maintenance processes, procedures, specific instructions, fault finding, and diagnostics efficiently. As before, status icons and feature names were also selected to be representative of the actual devices under control (icons are simplistic, informative, and familiar). Software diagnostic features assist the users and reduce diagnosis efforts by stakeholders. The web interfaces for management of the actual pivot, its maintenance, and other support functions were designed on a similar philosophy, namely to be familiar to the users with regard to their respective ontologies;
- **Heuristic 7 – Balance between agility and efficacy:** Users are allowed to be flexible in the operations and maintenance process (adjusting irrigation schedules, irrigation settings, etc.), but still follow a structured approach guided by realistic options from the requirements analysis, rather than allowing uncontrolled freedom. Besides the different options for setup, the controller has a range of available features that address all operational conditions. Remote (over-the-air) configuration and software updates are possible to simplify both end-user and engineering access efforts. The philosophy is thus to use IoT connectivity to provide the end-user with a variety of validated configuration and operations options, the maintenance personnel with advanced information to plan service routes and tasks, and engineering personnel the ability to change software remotely by following a structured process;

- **Heuristic 8 – Complexity reduction (Occam's razor)**: By using a layered approach, it was possible to limit complexity in each of the operational (day-to-day usage) and technical (engineering and maintenance) layers by design. Tasks were simplified by limiting the number of steps in processes, the amount of information on each display, including on-site LCD and web pages, and the actions to be taken by the operator. The controller's static display (fixed function) provides critical information at a glance. Operational functions that may be automated were included. For maintenance and support, automated fault detection and reporting were included, and tools such as diagnostics software further simplify the operations and maintenance processes;
- **Heuristic 9 – Validation, verification, and configuration management:** After installation, a user acceptance test is conducted to validate CPIS functionality on site. Additionally, feedback of system status is done after completion of a function and error reporting is done by means of escalation lists. That is, the loop is closed for each function, including utility failures (three-phase lines are individually checked, water pressure is checked, the CPIS mechanical structure is checked, and controller functions are checked). A clear indication of system errors is provided, and steps are provided in the manual as well as online, including options to further download system logs (by maintenance and engineering personnel). On-site verification can be seen on the controller LCD screen, and the mobile web allows remote verification and change. The fact that, for example, a user (farmer or maintenance technician) can remotely see exactly which CPIS tower is faulty allows actions to be planned in advance as some CPIS towers extend to 800m in a cornfield. This simplifies fault finding and verification, repair activities, planning (replacement stock to be taken to site) and other operational efforts for farmers and field support personnel;
- **Heuristic 10 – Support and communication:** Remote communication to CPIS is done using cellular networks, which may be limited in remote areas, but can be achieved using local networks (Wi-Fi on the farm via Ethernet) that are in turn connected to the cloud. The engineering team provided a managed network, which implies that any faults on the network can be immediately located. This crucial part of the IoT network plays a critical role in supporting this IoT system, as a managed and connected edge device forms the core of any IoT solution. Access to support for the farmer is via the installer/distributor in this case. In terms of specialised support, CPIS operations specialists, as well as the engineering team, provide support in the form of technically specialised operations, maintenance, and management knowledge. General support is provided by the installer and system integrator, in this case. The user and technical manuals were also found to be highly supportive.

Summary:

The following is a summary of the main findings from the analysis done on operations and maintenance:

- The focus on human-machine interfaces in the operational phase is customary, as confirmed by the literature study in Chapter 4. However, the addition of full remote capability in the form of a managed communications network allows remote access capability (including the principle of mobility, which falls under agility), which was not specifically listed as a usability heuristic – this is an addition to usability when working with IoT systems. Although this function falls under "communication", the remote access capability aspect is important in its own right as a key differentiator;
- Informative user interfaces were designed according to the principles of Nielsen's heuristics. In addition, the consistency provided by using web-based interfaces across different platforms is noteworthy. Simplification of interfaces by implementing static and dynamic screens was done to improve consistency and was based on constructivism, specifically with reference to the end-user ontology;
- Complexity reduction and layered thinking resulted in interfaces and functions limited to specific layers in the hierarchy and thus increased usability. Functions were designed and implemented on respective layers and user interaction defined on appropriate layers based on the ontology of the users on each layer. Thus no unnecessary information or control was present on a layer – no technical information was found on operational interfaces unless absolutely necessary. For example, the operational information on the LCD user interface related directly to agriculture, and technical indicators such as utility failures was presented simplistically using colours to indicate Go/No-Go conditions. Similarly, the LDC screen may be used for maintenance purposes, where technical information will then be displayed to a technician. The same approach was followed for web interfaces;
- Closing the loop through verification was found to be highly informative and usable as the system could automatically check for successful execution of functions on lower layers and report on failures to appropriate system resources. This is a form of automation not always considered in designs, but the availability of a communications network (IoT network) allows this functionality.

5.5.3.6 Product Retirement phase

Overview: Due to the limited time period over which this research was conducted, product retirement had not taken place when this document was compiled. It would thus be unscientific

to discuss this phase as part of the case study, and the life cycle usability model will be based on the information obtained from literature alone. However, the design of the system was done to support usability in this phase, as the decommissioning of hardware (in the field) is defined. In addition, should the system be re-engineered, all product data, including specifications and product life cycle data, will be available for a next cycle of development.

5.5.4 Product usability

As part of the overall system life cycle heuristics, the design and implementation of the CPIS product are highly important. Although the focus of this research is not on the actual CPIS system design, it is instructive to summarise the design philosophy. The CPIS controller and mobile web application were developed according to product specifications obtained in the Requirement Analysis phase. Nielsen's usability heuristics were used for product usability, and each heuristic is discussed as it was applied to the CPIS.

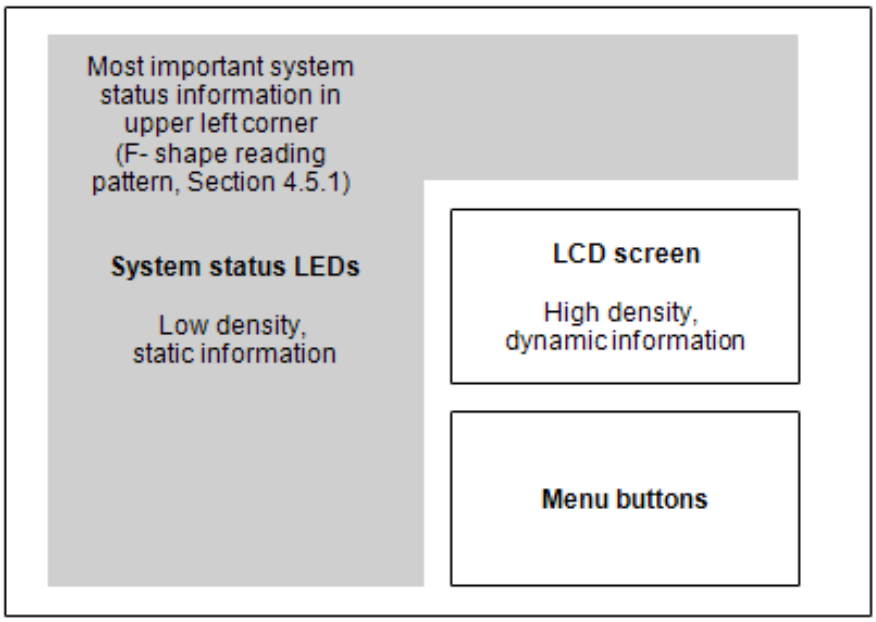

Figure 34: CPIS controller user interface usability

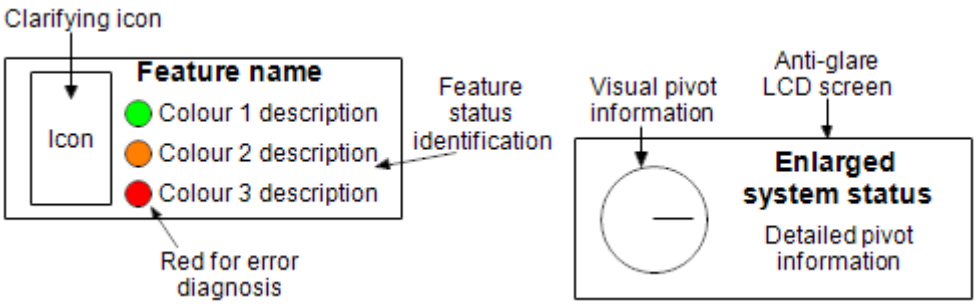

Figure 35: a) LED status indicator, b) LCD screen

- **Heuristic 1 – Visibility of system status:** End users can see the system status on the pivot controller LEDs, LCD screen, and on a mobile web application. In Figure 34, the basic layout of the controller user interface is shown. With the most important information in one location, the user can easily observe system operational and failure (error) status. Figure 35a shows the information shown for the LED status indicators. The combination of the icon, feature name, and colour clarification index simplifies feature identification and status interpretation. Figure 35b shows the LCD screen information layout, with the enlarged system status adding to system status visibility. Anti-glare, high contrast LCD screens were used for improved visibility in the field from an ergonomic perspective.
- **Heuristic 2 – Match between system and the real world:** Contextualisation was a particular challenge because of limited user literacy, while other users may have limited proficiency in the use of technology. The use of colours and representative icons is a powerful contextualisation tool aiding text-based information. As usual, red was used to indicate system errors (Figure 35a). The use of icons was incorporated for identification purposes, leading to limited misinterpretation. Web pages were designed to be representative of the environment with geographical information showing the actual CPIS environment in a GIS representation, overlayed with the pivot status.
- **Heuristic 3 – User control and freedom:** The user interfaces were designed for easy navigation and to give the user the freedom of selecting validated options. The screen information and menu were designed to navigate to the desired screen with the least effort (also reducing complexity) by navigating through the shortest paths. A Sector button, for example, allows the user to go directly to the Sectors menu from the main screen, which is an important function in pivot controllers.
- **Heuristic 4 – Consistency and standards:** Farmers replacing an existing controller most probably have owned another model and make from a different manufacturer. Thus, consistency with other existing controller products was implemented. For instance, the pushbuttons for the Mode, Direction, Start, Stop, etc. functions were designed to be consistent with other products, based on market research. Existing design, hardware, software, safety, and usability standards were followed in the design process. By applying Nielsen's heuristics, a standard was thus followed.
- **Heuristic 5 – Error prevention:** Users with limited technical capability have increased error rates when using IoT equipment as opposed to later generations. User less fluent in English can make use of icons to assists with feature/function identification, by design. The use of colours also assists in interpreting a function's status correctly. As

it is potentially dangerous to bypass a system, the controller was designed to prevent such tampering.

- **Heuristic 6 – Recognition rather than recall:** Static placement of critical icons and status indicators assist the user in recognising features, settings, and options, rather than having to memorise it. Standardised icons were used and can be easily recognised, but custom-designed icons were used in cases where a standard icon was not available. Other applied user interface design techniques include the use of standard names/headings and by using applicable. A similar approach was followed on the web page designs.
- **Heuristic 7 – Flexibility and efficiency of use:** Settings can easily be changed on the controller user interface as well as the mobile web application. The menu was designed to save time and guide the user with a logical grouping of information. Inefficient dials (which was customary) were replaced with simplified menus using Up, and Down buttons, a Backspace button, and the Enter button allow for easy selection and saving.
- **Heuristic 8 – Aesthetic and minimalist design:** The user interfaces are simplistically designed to avoid confusion, with the placement of LEDs, buttons, maps, etc. done to be aesthetically pleasing and easy to navigate. By reducing the number of screens (from static status indicators) and using simplified user input methods, the product is aesthetically pleasing and minimalistic.
- **Heuristic 9 – Help users recognise, diagnose, and recover from errors:** Users are guided in identifying, diagnosing and recovering from errors. On the display, the colours red and green were used for this purpose. The LCD screen and mobile web application contain additional diagnostic information, such as power levels, sensory information, tower locations, etc. Farmworkers are known to bypass the system override when not empowered to address errors, and remote monitoring was used to assist in this regard.
- **Heuristic 10 – Help and documentation:** The technical and user manuals have informative diagrams used in conjunction with numbered instructions. The technical training manual assists specialists and field support personnel - the user manual guides the user to operate the pivot controller efficiently and accurately. Appropriate user ontologies and diagrams are powerful tools in aiding text-based information. Online assistance, both through the IoT network and by conventional means, was made available to monitor systems and assist with system configuration, providing operational support, and providing technical support.

Shown below are images of the CPIS towers for a small installation, showing the control unit and interface, as well as a screenshot of the web page as discussed above.

Figure 36: CPIS controller and towers.

Figure 37: CPIS control unit with interface shown.

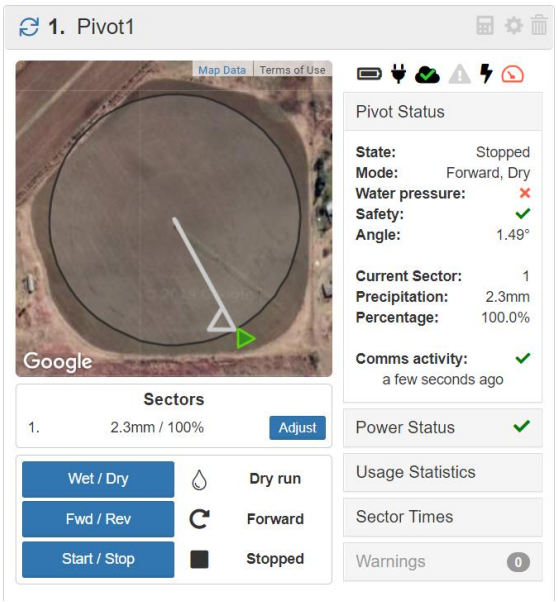

Figure 38: CPIS web page interface (mobile).

5.6 Action Design Research reflection on CPIS

In the ADR context, it is imperative to reflect upon the analysis conducted on the CPIS in the case study above. The following was observed:

1. MBSE is a good alternative to the document-based systems often used during requirements analysis. Furthermore, its value was confirmed in the design, implementation and testing phase – the MBSE models (and all its meta-data) support visual forms of communication, layered thinking (complexity reduction), configuration management, constructivism and finally, contextualisation;
2. The role of the Systems Engineer as virtually all role players (in the CPIS case study) interact with the Systems Engineer;
3. Rapid prototyping is most helpful in a constructivistic way to ensure product characteristics (to match requirements), validation and verification, and technical risk management are addressed by a prototype (also a form of model, in a general sense);
4. A good project manager, with applicable tools, supports communication and collaboration, which makes the design, implementation and verification process usable to all participants;
5. The iterative and incremental development approach (where functionality is specified initially and added iteratively), supported by rapid prototyping and support tools (such as CAD for modelling), also supported constructivism and verification;
6. The challenge of usability during manufacturing is less pronounced than for the initial life cycle phases due to the amount of work already done on usability in manufacturing systems, and the controlled nature of the environment;
7. The manufacturing datapack plays the most important role to ensure usability in the production process. This is in addition to existing human factors already present in the manufacturing environment;
8. The addition of remote access (via a cellular communications network) was critical in Manufacturing, as well as during Operations and Maintenance. The ability to configure, diagnose, and monitor systems during production were found to be highly supportive and is a heuristic specific to IoT systems;
9. Informative user interfaces were implemented using Nielsen’s heuristics. The consistency provided by using web-based interfaces across different platforms is important and is also a key component of IoT systems;
10. Closing the loop through automated, remotely executed verification in real-time is a form of automation not always considered in general system designs, and the availability of an IoT network-enabled this feature, which should be heuristic in IoT system design.

The above considerations, following from a process of reflection, add value to the definition of usability in a system's full life cycle. It is instructive to see that not only Systems Engineering in general, but specifically model-based engineering, adds to usability for the reasons shown above. Rapid prototyping is also a form of modelling and confirms the observation that MBSE is useful in this context. It is generally known that a good project manager is critical to delivering on time, within budget and within resource constraints, which was also observed in this case study, but with a usability focus. In addition to the "usual" usability considerations during manufacturing and operation, it was noted that IoT systems actually add significant value to usability due to their interconnected nature. The connected nature allowed remote access to devices through final stages of production, during integration, deployment, operation and also (importantly) maintenance and upgrades. The network also allowed automated verification, which is important, specifically during operation.

6. Validation and conclusion

6.1 Validation

The study developed a full system life cycle usability framework for IoT systems. Inside the DSR paradigm, the ADR method was used to conduct this research, supported by the quantitative risk management (QRM) method for quality management. The research challenges (as identified in Chapter 3 and seen in Table 5) are addressed by the five research solutions (discussed in Section 3.2.2) also shown in Table 5.

To address the lack of a clear definition of IoT in general, a definition was created from literature. The new definition expands on previous IoT definition attempts to include i) the full system life cycle, and ii) all system users and stakeholders. This definition of IoT forms the foundation for the usability framework in research solutions one and three.

To address the lack of information in usability in IoT and the full IoT system life cycle, a usability framework was developed. The usability framework is validated through i) literature, ii) an IEEE article, and iii) the centre pivot irrigation system (CPIS) case study.

i. **Literature:** From literature, general usability issues for IoT systems were identified. After the two sets of usability heuristics (Nielsen and generalised) had been obtained, the general life cycle issues were recalled, and the general life cycle heuristics were used to show that they may be used to address the issues. This validates the applicability of all the usability heuristics in a general sense;
ii. **IEEE article:** An article on the usability framework discussed in point was compiled and accepted. The article was peer-reviewed and accepted by IEEE Africon 2019. The article is available in Appendix A, as well as the acceptance email;
iii. **CPIS case study:** The usability framework was applied to the development of a CPIS. During the case study, both sets of usability heuristics were applied to the system and product. The usability heuristics improved usability in the Requirements Analysis to Operations and Maintenance phases (as discussed in Section 5.5.3) and were accepted by the pivot vendor. The pivot vendor also had significant orders for the pivot controller on its books, indicating public acceptance.

It was seen that a valid usability baseline could be developed and implemented through the proper application of systems engineering, complexity reduction, contextualisation, constructivism, and efficient communication, as contained in the usability heuristics. Usability

issues not sufficiently addressed by the usability heuristics were identified, and additional usability heuristics were identified to address those issues.

Table 5: Solution validation

RESEARCH CHALLENGES	1) Lack of information on usability in IoT	2) No clear definition of IoT in general	3) Usability over the full life cycle in IoT systems not defined	4) Usability in pivot irrigation controllers not fully applied	5) Focusses mostly on end-user usability
RESEARCH SOLUTIONS	1) Create framework for usability in IoT systems	2) Give a clearer definition of IoT	3) Create framework for usability over full life cycle	4) Usability applied to irrigation pivot controller - ADR case study	5) Expand focus to all IoT system stakeholders
Clear definition of IoT in general		↑			↑
Artefact 1: Usability framework for full system life cycle (generalized heuristics and ergonomics)	↑	↑	↑		↑
Case study (Artefact 2): Center pivot irrigation system with interface designs as deliverable and a gap analysis	↑		↑	↑	↑
IEEE conference publication	↑		↑		

6.2 Conclusion

Usability issues, such as inadequate access to information, are identified in literature and experienced by system users and stakeholders. These usability issues are present in all IoT system life cycle phases and indicate a general lack of usability in IoT systems over all system life cycle phases. From literature, it was also seen that there is no clear definition for IoT in

general and that the focus is mainly on end-user usability. A general lack of usability over all system life cycle phases and an end-user focus results in inefficient IoT systems and costly system issues. There are numerous system processes, users, and stakeholders that should be addressed in system designs and management, respectively.

A clearer definition of IoT was developed, which is defined over the entire system life cycle and expands to all systems users and stakeholders. The phases for the full life cycle are grouped as 1) Requirements Analysis (RA), 2) Design, Implementation, and Testing (DIT), 3) Product Manufacturing (PM), 4) Operations and Maintenance (O&M), and 5) Product Retirement (PR).

Using the above IoT definition, a life cycle usability framework for IoT systems was developed. Research regarding existing methods for improved usability in products and systems indicated that the most comprehensive set of usability heuristics is Nielsen's usability heuristics. As Nielsen's heuristics are primarily used for product user interfaces, a generalised set of life cycle heuristics for system usability was created and validated from Nielsen's heuristics. An IEEE article was published for peer review on the usability framework [118], for which the authors were commended.

Research on general usability issues over a system life cycle was used to show that the usability heuristics are applicable to product (Nielsen) and system (generalised) usability issues. The list of issues is not comprehensive, and there may be additional heuristics required, specifically at a managerial level. However, this is the first iteration of research conducted in the spirit of ADR and shows that an artefact could be designed by using a valid usability heuristic baseline.

The usability framework was applied to the development of a CPIS. A functional analysis of the CPIS was conducted to identify system activities, users, stakeholders, and interfaces. Operational specialists assisted with this. All usability heuristics were applied to the CPIS system (generalised) and product (Nielsen). The system designs include PM, O&M, and PR factors for system efficiency. The system design was validated and accepted by the pivot vendor (client).

During requirements acquisition, issues with the previous versions of CPIS systems and products were identified. Combined with the general usability issues that apply to the CPIS, the applied usability heuristics shows that a CPIS could be designed by using a valid usability heuristic baseline.

In conclusion, usability allows developers to function more efficiently, with the use of appropriate resources. The generalised usability heuristics relate closely to system engineering process functions. Heuristics such as system representation, quality and requirements management, and technical risk analysis are all functions that relate to systems engineering principles. Contextualisation, constructivism, complexity reduction, and agility are all additional principles and are used in conjunction with systems engineering for a more efficient system.

This study shows that applying proper systems engineering functions, as well as efficient contextualisation, constructivism, complexity reduction, and communication principles improve the usability of a system. Future research may introduce more factors, but this research shows that the usability baseline is a valid usability framework for IoT systems.

6.3 Recommendations

In this study, research was done on general issues linked to system usability, but the list is not comprehensive. Further studies on general and specific usability issues may indicate issues that may require additional heuristics. This will increase the value of the usability heuristics baseline, as system usability is improved yet again.

The usability framework was applied to the CPIS development, but applying the framework to more systems will add to the framework validation, as well as indicating existing gaps in the current usability framework. The additional systems should include systems in other sectors, such as security, transportation, etc. Examples of such systems are surveillance systems, vehicle electronics systems, and others.

Including questionnaires in the following case studies will provide additional validation and gap analysis from other information sources, such as end-users, production personnel, field support personnel, etc. Questionnaires were not included in this study as the verification from literature, peer-review, and an accepted physical product were considered sufficient evidence of validity.

This study focused on technical aspects. Still, other aspects, such as business and social aspects, are in many cases just as important in system usability and should be included in following research. Social and business issues are, in many cases, the root cause of technical issues, such as inefficient communication, resistance to change, language, culture discrepancies, and many more challenges facing development companies.

7. REFERENCES

[1] A. Bhattacherjee, *Introduction to Research, Social Science Research: Principles, Methods, and Practices*. 2012.

[2] A. R. Hevner, “A Three Cycle View of Design Science Research,” *Scand. J. Inf. Syst.*, vol. 19, no. 2, 2007.

[3] A. Hevner, S. March, J. Park, and S. Ram, “Design Science Research in Information Systems,” *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004.

[4] Lindgren, Sein, Henfridsson, Rossi, and Purao, “Action Design Research,” *MIS Q.*, vol. 35, no. 1, p. 37, 2017.

[5] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2008.

[6] B. S. Blanchard and W. J. Fabrycky, *Systems Engineering and Analysis*, Third. New Jersey: Prentice Hall, 1998.

[7] T. M. Papp, “CrashAppTM –Concurrent Multiple Stakeholder Evaluation of a DSR Artefact,” no. October, p. 136, 2017.

[8] M. T. Mullarkey and A. R. Hevner, “New Horizons in Design Science: Broadening the Research Agenda,” *Desrist 2015*, vol. LNCS 9073, pp. 121–134, 2015.

[9] W. Quesenbery, “What Does Usability Mean: Looking Beyond ‘Ease of Use’ - Whitney Interactive Design,” *WQUsability*. pp. 1–8, 2015.

[10] International Organization For Standardization, “ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability.” 1998.

[11] I. D. Foundation, “Usability,” *Interact. Des. Foundadtiontion*, 2013.

[12] J. Nielsen, “10 Usability Heuristics for User Interface Design,” *Nielsen Norman Gr.*, p. 1, 1995.

[13] J. N. R. Molich, “Heuristic Evaluation of User Interfaces,” *CHI’90 Proc.*, Apr. 1990.

[14] J. Nielsen, “Enhancing the explanatory power of usability heuristics,” *Hum. Factors Comput. Syst.*, pp. 152–158, 1994.

[15] J. Folmer, Eelke; Bosch, “Architecting for Usability; a Survey,” *J. Syst. Softw.*, no. 70, pp. 61–78, 2004.

[16] M. Van Welie, G. C. Van Der Veer, and A. Eliëns, “Breaking down Usability,” *Proc. INTERACT*, pp. 613–620, 1999.

[17] O. L. Huye, “University’s official website design framework using human computer interaction rules,” Universiti Malaysia Pahang.

[18] ISO/IEC, “ISO 9126: Software Engineering - Software Product Quality - Part 1: Quality Model.” 2001.

[19] D. Zimmermann, “Universal Acess in Human Computer Interaction. Coping with Diversity,” vol. 4554, no. July 2007, 2007.

[20] S. Winter, S. Wagner, and F. Deissenboeck, “Engineering Interactive Systems 2008,” *Eng. Interact. Syst. 2008*, pp. 106–122, 2008.

[21] J. Nielsen, “Usability 101: Introduction to Usability,” *Nielsen Norman Gr.*, 2012.

[22] E. Kiliç Delice and Z. Güngör, “The usability analysis with heuristic evaluation and analytic hierarchy process,” *Int. J. Ind. Ergon.*, vol. 39, no. 6, pp. 934–939, 2009.

[23] W. Quesenbery, “What Does Usability Mean: Looking Beyond ‘Ease of Use,’” *WQUsability*, pp. 1–8, 2014.

[24] B. Caldwell, C. Michael, L. G. Reid, and G. Vanderheiden, “Web Content Accessibility Guidelines (WCAG) 2.0,” *W3C*, 2008. [Online]. Available: https://www.w3.org/TR/2008/REC-WCAG20-20081211/#intro-layers-guidance.

[25] D. J. Mayhew, *The Usability Engineering Lifecycle: a practitioner’s handbook for user interface design*. Morgan Kaufmann Publishers, 1999.

[26] O. Isaksson, T. C. Larsson, and A. Ö. Rönnbäck, “Development of product-service systems: challenges and opportunities for the manufacturing firm,” *J. Eng. Des.*, vol. 20, no. 4, pp. 329–348, Aug. 2009.

[27] L. Ballard, G.; Koskela, “On the agenda of design management research,” in

Proceedings of the International Conference on Lean Construction, Guaruja, Brazil, 1998.

[28] J. Kelanti, M., Hyysalo, P. Kuvaja, M. Oivo, and A. Välimäki, “A Case Study of Requirements Management: Toward Transparency in Requirements Management Tools,” in *Proceedings of the Eighth International Conference on Software Engineering Advances (ICSEA 2013)*, 2013, pp. 597–604.

[29] J. M. Bhat, M. Gupta, and S. N. Murthy, “Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing,” *IEEE Softw.*, vol. 23, no. 5, pp. 38–44, Sep. 2006.

[30] M. Weber and J. Weisbrod, “Requirements engineering in automotive development-experiences and challenges,” in *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2003, pp. 331–340.

[31] J. A. Bubenko, “Challenges in requirements engineering,” in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE’95)*, 1995, pp. 160–162.

[32] L. Lehtola, M. Kauppinen, and S. Kujala, “Requirements Prioritization Challenges in Practice,” no. May 2014, 2004, pp. 497–508.

[33] B. Boehm and R. Turner, “Management Challenges to Implementing Agile Processes in Traditional Development Organizations,” *IEEE Softw.*, vol. 22, no. 5, pp. 30–39, Sep. 2005.

[34] R. J. Chapman, “The controlling influences on effective risk identification and assessment for construction design management,” *Int. J. Proj. Manag.*, vol. 19, no. 3, pp. 147–160, Apr. 2001.

[35] A. Griffin and J. R. Hauser, “Patterns of Communication Among Marketing, Engineering and Manufacturing—A Comparison Between Two New Product Teams,” *Manage. Sci.*, vol. 38, no. 3, pp. 360–373, Mar. 1992.

[36] P. Tzortzopoulos and C. Formoso, “Considerations on application of lean construction principles to design management,” *Proc. IGLC-7*, pp. 335–344, 1999.

[37] R. Owen *et al.*, “Challenges for integrated design and delivery solutions,” *Archit. Eng. Des. Manag.*, vol. 6, no. SPECIAL ISSUE, pp. 232–240, 2010.

[38] L. Koskela, P. Huovila, and J. Leinonen, “Design management in building construction:

From theory to practice," *J. Constr. Res.*, vol. 03, no. 01, pp. 1–16, Mar. 2002.

[39] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[40] F. Tao, Y. Wang, Y. Zuo, H. Yang, and M. Zhang, "Internet of Things in product life-cycle energy management," *J. Ind. Inf. Integr.*, vol. 1, pp. 26–39, Mar. 2016.

[41] V. A. Mabert and M. A. Venkataramanan, "Special Research Focus on Supply Chain Linkages: Challenges for Design and Management in the 21st Century," *Decis. Sci.*, vol. 29, no. 3, pp. 537–552, Jul. 1998.

[42] J. C. Knight, "Safety critical systems," in *Proceedings of the 24th international conference on Software engineering - ICSE '02*, 2002, p. 547.

[43] W. J. Glantschnig, "Green design: an introduction to issues and challenges," *IEEE Trans. Components, Packag. Manuf. Technol. Part A*, vol. 17, no. 4, pp. 508–513, 1994.

[44] S. Takata *et al.*, "Maintenance: Changing Role in Life Cycle Management," *CIRP Ann.*, vol. 53, no. 2, pp. 643–655, 2004.

[45] A. Meissner, T. Luckenbach, T. Risse, T. Kirste, and H. Kirchner, "Design Challenges for an Integrated Disaster Management Communication and Information System," *First IEEE Work. Disaster Recover. Networks*, no. DIREN 2002, 2002.

[46] J. P. Liyanage and U. Kumar, "Towards a value-based view on operations and maintenance performance management," *J. Qual. Maint. Eng.*, vol. 9, no. 4, pp. 333–350, Dec. 2003.

[47] K. L. Chin, J.P.; Diehl, V.A.; Norman, "Development of an instrument measuring user satisfaction of the human-computer interface," *Proc. CHI'88 Conf. Hum. Factors Comput. Syst.*, pp. 213–218, 1988.

[48] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Q.*, vol. 13, no. 3, p. 319, 1989.

[49] R. J. Lewis, "IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. Boca Raton, FL: Human Factors Group," *IBM Tech. Rep.*, vol. 54, no. 1, p. 786, 1993.

[50] J. Dul *et al.*, "A strategy for human factors/ergonomics: Developing the discipline and

profession," *Ergonomics*, vol. 55, no. 4, pp. 377–395, 2012.

[51] INCOSE, "Systems Engineering," *Systems Engineering*, 2010. [Online]. Available: https://www.incose.org/systems-engineering.

[52] M. C. Davis, R. Challenger, D. N. W. Jayewardene, and C. W. Clegg, "Advancing socio-technical systems thinking: A call for bravery," *Appl. Ergon.*, vol. 45, no. 2 Part A, pp. 171–180, 2014.

[53] P. Carayon, "Human factors of complex sociotechnical systems," *Appl. Ergon.*, vol. 37, no. 4 SPEC. ISS., pp. 525–535, 2006.

[54] J. R. Wilson, "Fundamentals of systems ergonomics/human factors," *Appl. Ergon.*, vol. 45, no. 1, pp. 5–13, 2014.

[55] A. Dillon, "Group dynamics meet cognition : applying socio- technical concepts in the design of information systems," *New SocioTech Graffiti Long Wall*, pp. 119–126, 2000.

[56] R. Jiang, *Introduction to Quality and Reliability Engineering*, XXII. Berlin: Springer, 2015.

[57] R. G. Du Preez, "Deployment, re-engineering and risk analysis of a C-band weather radar to build local capacity in South Africa," North-West University, 2017.

[58] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2016-March, pp. 3928–3937, 2016.

[59] A. Gilchrist, *Industry 4.0: The Industrial Internet of Things*. New York: Springer Science+Business Media, 2016.

[60] V. Alcácer and V. Cruz-Machado, "Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems," *Eng. Sci. Technol. an Int. J.*, no. xxxx, 2019.

[61] E. A. Lee, "Cyber physical systems: Design challenges," *Proc. - 11th IEEE Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput. ISORC 2008*, pp. 363–369, 2008.

[62] A. Radziwon, A. Bilberg, M. Bogers, and E. S. Madsen, "The smart factory: Exploring adaptive and flexible manufacturing solutions," *Procedia Engineering*, vol. 69. pp. 1184–1190, 2014.

[63] S.-H. Yang, *Wireless Sensor Networks*. London: Springer London, 2014.

[64] G. Koschnick, M. Hankel, and B. Rexroth, “RAMI 4.0-Structure The Reference Architectural Model Industrie 4.0 (RAMI 4.0) Contact: Reference Architectural Model Industrie 4.0,” vol. 0, no. April, 2015.

[65] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, “Industry 4.0 – An Introduction in the phenomenon,” *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[66] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[67] N. K. Mukhopadhyay, S.C.; Suryadevara, *Internet of Things*, 9th ed., vol. 9. Cham: Springer International Publishing, 2014.

[68] D. Kiritsis, “Closed-loop PLM for intelligent products in the era of the Internet of things,” *CAD Comput. Aided Des.*, vol. 43, no. 5, pp. 479–501, 2011.

[69] H. C. Y. Chan, “Internet of Things Business Models Internet of Things (IoT), Business Model, Strategy and Tactic in IoT, Case Studies in IoT,” *J. Serv. Sci. Manag.*, vol. 8, no. 4, pp. 552–568, 2015.

[70] L. Atzori, A. Iera, G. Morabito, and M. Nitti, “The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization,” *Comput. Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[71] J. Peschke, A. Luder, and H. Kuhnle, “The PABADIS’PROMISE architecture - a new approach for flexible manufacturing systems,” *IEEE*, vol. 1, pp. 491–496, 2006.

[72] IEA, *Definition and Domains of Ergonomics*. .

[73] J. Dul and W. P. Neumann, “Ergonomics contributions to company strategies,” *Appl. Ergon.*, vol. 40, no. 4, pp. 745–752, 2009.

[74] B. M. Kleiner, “Macroergonomics: Analysis and design of work systems,” *Appl. Ergon.*, vol. 37, no. 1 SPEC. ISS., pp. 81–89, 2006.

[75] B. T. Karsh, P. Waterson, and R. J. Holden, “Crossing levels in systems ergonomics: A framework to support ‘mesoergonomic’ inquiry,” *Appl. Ergon.*, vol. 45, no. 1, pp. 45–54, 2014.

[76] H. Haines, J. R. Wilson, P. Vink, and E. Koningsveld, “Validating a framework for participatory ergonomics (the PEF),” *Ergonomics*, vol. 45, no. 4, pp. 309–327, 2002.

[77] IBM Corporation, “Common User Access (CUA) guidelines,” *IBM Knowledge Center*, 2014. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.f54dg00/cuahlp.htm.

[78] S. J. Dorey, “Common User Access -- Principles of GUI Design,” *Princ. User Interface Des.*, pp. 1–11, 2007.

[79] Microsoft, “User Interface Principles - Windows applications _ Microsoft Docs,” *Windows Dev Center*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/desktop/appuistart/-user-interface-principles. [Accessed: 14-Apr-2019].

[80] ISO, “ISO 9241-14:1997 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 14: Menu dialogues,” 1997. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-14:ed-1:v1:en.

[81] ISO/IEC, “ISO 11581-3, Information technology — User system interfaces and symbols — Icon symbols and functions — Part 3: Pointer icons,” 2000. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en.

[82] ISO/IEC, “ISO_IEC 11581-10_2010 - Information technology -- User interface icons -- Part 10_ Framework and general guidance,” 2010. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en.

[83] I. Apple Computer, *Macintosh Human Interface Guidelines*. Addison-Wesley Publishing Company, 1995.

[84] Interaction Design Foundation, “What is User Interface (UI) Design? | Interaction Design Foundation.” p. 1, 2018.

[85] K. Pernice, “F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant (Even on Mobile),” *NNGroup*, 2017. [Online]. Available: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/. [Accessed: 15-Apr-2019].

[86] KDE Human Interface Guidelines, “KDE Human Interface Guidelines — Human Interface Guidelines documentation,” *KDE Human Interface Guidelines*. [Online].

Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en. [Accessed: 15-Apr-2019].

[87] A. Phocaides, “Ch 10: The center pivot irrigation system,” in *Pressurized Irrigation Techniques*, Second., Rome: FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, 2007, pp. clix–clxxvi.

[88] Senter360, “Senter360,” *Senter360*, 2017. [Online]. Available: http://www.senter360.co.za/. [Accessed: 27-Feb-2019].

[89] Global Beef, “Sustainable, Profitable Agriculture.” .

[90] Irrigation.Education Community, “How a Center Pivot Irrigation Machine Works,” *irrigation.education*, 2016. [Online]. Available: http://blog.irrigation.education/blog/how-a-center-pivot-works. [Accessed: 09-Jul-2019].

[91] X. Dong, M. C. Vuran, and S. Irmak, “Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems,” *Ad Hoc Networks*, vol. 11, no. 7, pp. 1975–1987, 2013.

[92] C. R. Camp, E. J. Sadler, D. E. Evans, L. J. Usrey, and M. Omary, “Modified Center Pivot System for Precision Management of Water and Nutrients,” *Appl. Eng. Agric.*, vol. 14, no. 1, pp. 23–31, 1998.

[93] Valley, “ICON10.” pp. 1–2, 2018.

[94] Valley, “ICON5.” pp. 1–2, 2018.

[95] Valley, “ICON1.” pp. 1–2, 2018.

[96] Valley, “ICONX.” pp. 1–2, 2018.

[97] Talbert, “Talbert RAIN User & Installer Manual.” pp. 1–24.

[98] Agrico, “Advanced rain user guide.” Agrico, pp. 1–9.

[99] Agrico, “Agrico spilpuntbeheerstelle.” Agrico, p. 100.

[100] Zimmatic, “Irrigation control field management products.” Lindsay Cooperation, Omaha, pp. 1–4, 2017.

[101] Linsay Cooperation, “FIELDNET Irrigated remote irrigation management.” Lindsay

Cooperation, Omaha, pp. 1–10, 2018.

[102] Lindsay Cooperation, “FIELDNET Pivot Control & FIELDNET Pivot Control Lite.” Lindsay Cooperation, Omaha, pp. 1–4, 2018.

[103] Automation Products, “Main Panel One Tower 110V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots110-main-panel-one-tower-110v.html. [Accessed: 27-Feb-2019].

[104] Automation Products, “’Main Panel One Tower 220V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots220--main-panel-one-tower-220v.html. [Accessed: 27-Feb-2019].

[105] Automation Products, “Main Panel One Tower 24V DC,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots24-main-panel-one-tower-24v-dc.html. [Accessed: 27-Feb-2019].

[106] Automation Products, “Main Panel One Tower Inverter 220V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-poti-main-panel-one-tower-inverter-220v.html. [Accessed: 27-Feb-2019].

[107] Automation Products, “Main Panel Two Tower 110V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-ptt110-main-panel-two-tower-110v.html. [Accessed: 27-Feb-2019].

[108] Automation Products, “Main Panel Two Tower 220V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots220--main-panel-one-tower-220v.html. [Accessed: 27-Feb-2019].

[109] Automation Products, “Main Panel Large 110V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl110-main-panel-large-110v.html. [Accessed: 27-Feb-2019].

[110] Automation Products, “Main Panel Large 220V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl220-main-panel-large-220v.html. [Accessed: 27-Feb-2019].

[111] Automation Products, “Main Panel Large Inverter 220V,” *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl220it-main-panel-large-inverter-220v.html. [Accessed: 27-Feb-2019].

[112] M. Abdel-Basset, G. Manogaran, and M. Mohamed, “Internet of Things (IoT) and its impact on supply chain: A framework for building smart, secure and efficient systems,” *Futur. Gener. Comput. Syst.*, vol. 86, pp. 614–628, 2018.

[113] J. E. W. Holm, “IIoT - Systems Approach.” Potchefstroom, South Africa, 2018.

[114] F. N. Akhras and J. A. Self, “System Intelligence in Constructivist Learning,” *Int. J. Artif. Intell. Educ.*, vol. 11, pp. 344–376, 2000.

[115] J. A. Estefan, “Survey of Model-Based Systems Engineering (MBSE),” *INCOSE MBSE Initiat.*, 2008.

[116] A. Baker, “Simplicity,” in *Stanford Encyclopedia of Philosophy*, Winter 201., E. N. . Zalta, Ed. Metaphysics Research Lab, Stanford University, 2016.

[117] F. Nayebi, J. M. Desharnais, and A. Abran, “An expert-based framework for evaluating iOS application usability,” *Proc. - Jt. Conf. 23rd Int. Work. Softw. Meas. 8th Int. Conf. Softw. Process Prod. Meas. IWSM-MENSURA 2013*, no. January 2014, pp. 147–155, 2013.

[118] J. E. W. Holm and E. Du Plessis, “Usability – a Full Life Cycle Perspective,” *IEEE Africon*, 2019.

[119] D. A. Viljoen, “Synthesis and evaluation of engineering processes for the development of airborne electronic equipment,” North-West University, Potchefstroom Campus, 2016.

[120] I. Yüksel, “Developing a Multi-Criteria Decision Making Model for PESTEL Analysis,” *Int. J. Bus. Manag.*, vol. 7, no. 24, 2012.

ANNEXTURE A – IEEE ARTICLE

Usability – a Full Life Cycle Perspective

Johann E.W. Holm [0000-0003-2862-909X]
Department of Computer and Electronic Engineering
North-West University
Potchefstroom, South Africa
Johann.Holm@nwu.ac.za

Esté du Plessis [0000-0002-8777-5399]
Department of Computer and Electronic Engineering
North-West University
Potchefstroom, South Africa
24059706@student.g.nwu.ac.za

***Abstract*—In this paper, the applicability of Nielsen's ten usability heuristics to a system full life cycle is investigated. The challenge is to define usability for Internet-of-Things (IoT) type systems in a full life cycle context, specifically for a product's enabling system. Usability in a full life cycle is to ensure all users of a multidisciplinary system are acknowledged, and not only end-users. Enabling systems are most often neglected when considering usability as focus is on the end-product. A baseline is first developed (in principle) by combining results from literature and is then generalized for a larger system. It was observed that usability heuristics can be generalized and applied to a system full life cycle. The resulting generalized heuristics consider system users such as clients, developers and other beneficiaries. The value of usability in the system development phase (before operation and maintenance) should be able to alleviate the challenges encountered during development, as a literature study verifies. It is interesting to see that systems engineering practices address many usability challenges, with contextualization, constructivism, complexity reduction, agility and communication confirmed as additional usability factors.**

Keywords—Usability, system, full life cycle, heuristics.

I. Introduction

Usability is often understood to be linked to interfaces between end-users and technology. This paper addresses usability over an Internet-of-Things (IoT) system full life cycle, which implies that users participating in both product acquisition and product utilization phases must be considered. People are involved in requirements analysis, concept, preliminary and detail design, integration and testing, production, operation and maintenance. Often, product issues during operation can be traced back to design shortfalls upstream. The question is: "What does usability in a full life cycle look like, and how can it be ensured?"

The focus on usability in this paper is purely on the technical aspects of systems. Project management aspects (project schedule, risk, cost and resources) are not included in this study. Also, the general notions of usability, such as ergonomics, safety and other human factors that apply to the actual product, are not specifically considered here as these are well defined in other texts. Rather, aspects of usability associated with the enabling systems are considered (i.e. systems to establish a product and to support and maintain that product in operation).

II. Research methodology

Without detracting from the end-goal of producing a usable product or equipment of a larger system, a *usability baseline* is required to show how usability will affect all system users in a product's full life cycle. To this end, research was conducted on the definition of usability in life cycle phases of a product and its enabling system (referred to as the "product" with the understanding that an enabling environment is always present). That is, usability must be defined from where requirements are analysed, through all design phases, product construction or production, operational deployment and eventually to the end of life decision to either upgrade or phase out. The end goal of such a baseline is to set a reference for further research and to show the value of usability from a system life cycle perspective.

A literature study was conducted to find suitable guidelines for usability. As the focus is mostly on IoT-type systems with a significant IT component, the study commenced with usability as defined for information systems. From this study, applicable heuristics were obtained, from which a generalised set of heuristics was created. As literature did not contain much information on full life cycle heuristics for enabling systems, the generalised heuristics were verified against

challenges (also from literature) found in different life cycle phases. Basic architectural analyses were conducted in life cycle phases to show the relevance of this study.

This study is a first effort to establish a larger framework for usability in a system full life cycle and is by no means comprehensive. However, the baseline created here will be elaborated upon in further studies to derive a full set of heuristics that apply to both the product and its enabling system.

III. Human factors in systems

A. Usability

Usability is an important design requirement for product and system development. ISO 9241-11 defines usability as "the extent to which a product can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction in a specific context of use" [10]. There are many definitions of usability. For Shackel, effectiveness, learnability, flexibility, and attitude are the dominant factors. For Nielsen, learnability, efficiency, memorability, errors, and satisfaction are dominant. ISO 9241-11 defines the dominant factors as effectiveness, efficiency, and satisfaction. ISO 9126-1 has understandability, learnability, operability, and attractiveness as dominant factors [15]. These factors are mostly used to define the end user's experience but could be generalised for all system users.

A popular usability testing method utilises usability heuristics. Popular usability heuristics are Nielsen's ten heuristics [12], [14], Shneiderman's eight golden rules of interface design and Norman's seven principles [22]. Shneiderman's rules are used in interface design [16]. In this study, usability is extended to cover a whole system, not just interface design for the end product. This is also the case for Norman's seven principles. Nielsen's ten heuristics are also mostly used for interface design, but in this study, Nielson's heuristics were selected as the most suitable to be generalised across the whole system full life cycle as these were found to be encompassing. Nielsen's ten usability heuristics [22] are discussed in Table I in a later section.

Nielson's ten heuristics are more applicable to information systems and may not be comprehensive when considering all life cycle phases – additional heuristics may be added to this baseline. However, the applicability of the abstracted (or rather, generalised) set of heuristics is of interest in this paper. It is also acknowledged that research presented here is not comprehensive and is a first attempt at defining a reference baseline. Further studies will elaborate on this baseline.

B. System full life cycle

According to the International Council of Systems Engineering (INCOSE):

"Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focusses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem." [51]

The following phases are typically present in a life cycle [6] as briefly discussed below:

- Definition of need: This phase includes feasibility studies about the product [120] to ensure validation of the need;
- Conceptual design: This phase is mostly concerned with requirements elicitation and development of system and development specifications [6]. Operational level system functional flows and architectures are defined in this phase;
- Preliminary design: Further analysis is done, and design requirements are allocated to system functional elements [6]. This phase is critical in system development as "design for" requirements are also allocated, including "design for usability" [15];
- Detail design: Amongst other activities, prototype development and testing are done. The detail design is done using specification documents and "design for" guidelines. Throughout, verification testing is done;
- Industrialization: Production prototypes are tested and evaluated. Preparation for manufacturing of the product is done, resulting in a manufacturing data pack [6];
- Production and distribution. Manufacturing is done with assembly, testing, packaging and distribution of products according to the manufacturing data pack specifications [56]. Warranties are also honoured through logistics channels;

- Operation and maintenance: Products are finally used by end-users and maintained and serviced by appropriate resources;
- Phase-out or retirement and disposal: Production is terminated, sales and support stop and retired units are recycled and disposed of [6].

In each of the above phases, there are four types of users and stakeholders [50]:

- System actors: employees, end-users, etc.;
- System experts: engineers, business consultants and analysts, domain specialists, technicians, etc.;
- System decision makers: managers, supervisors, director's board, etc.;
- Systems influencers: media, governments, workers unions, standardisation organisations, regulators, people with a public interest, etc.

For this paper, only the first two listed groups will be considered as users (to limit complexity) [50]. Regardless, the list shows that the term "user" should be considered in a broader sense than just end-users.

IV. Usability in a full life cycle

In order to define a usability baseline, it is necessary to consider different sources of information used to define such a framework. Existing literature resulted in the selection of Nielson's ten heuristics, and generalization thereof to form a baseline for general usability. In addition, from basic operational analysis, architectural diagrams were constructed for all life cycle phases, showing all actors and associated interfaces. Finally, from literature, human-related shortfalls of activities in life cycle phases were studied and were used to show where a framework can potentially alleviate some of the human-related issues associated with specific phases.

In order to simplify the usability baseline, life cycle phases were grouped to result in the following: 1) requirements analysis, 2) design, implementation, and testing, 3) product manufacturing, 4) operations and maintenance, and 5) retirement.

A. Nielson's generalised heuristics

Nielson's heuristics were generalised by translating them to apply to general systems that include people, equipment, tools etc. used in life cycle phases. The generalised matrix is presented in Table I with a discussion on each heuristic to show its relevance. It is important to note that Nielson's heuristics (in its original form) still apply to the actual products under development, while the generalised heuristics apply to the overall life cycle.

Process usability is the main research point for this study. Ten usability heuristics were derived for the analysis. They are generalised heuristics derived from Nielsen's ten usability heuristics [12]. Nielsen's ten heuristics and the derived generalised heuristics are shown in Table I.

B. Additional human factor considerations

In addition to the above heuristics (which are used mainly in information systems environments), physiological ergonomic factors and safety were considered as part of the usability baseline.

Safety is always an important factor in any project. Depending on the project type, safety factors may differ, but each project must comply with both local and international OHS standards and policies. Disregarding these standards and policies may lead to injuries, death, losses and associated legal challenges. Design for safety-critical systems is a discipline on its own, and the processes and methods used for airborne electronics may form part of a future baseline (kindly refer to [119] for an extensive treatment on this topic). A detailed discussion will not be provided here.

Ergonomics is a discipline in its own right [114] that concerns mostly physical ergonomics or rather anthropometric factors. This includes human body positions, vision, hearing, and touch. The physical position and size of the product may have an effect on the physical position of the body of the user. Tools and equipment used in all life-cycle phases must address the requirements for anthropometrics, which are well known in the design industry. Suffice to say that physical ergonomics include experiences for designers, manufacturers (when using physical tools etc.), installers and maintenance personnel amongst others. A detailed treatment will not be provided here as physical ergonomics are well-known and defined.

TABLE I. NIELSEN'S USABILITY HEURISTICS VS. DERIVED GENERALISED USABILITY HEURISTICS

#	Nielsen's ten heuristics	Generalised heuristics	Usability heuristic definitions
1	Visibility of system status	System representation and status	IS perspective: The current state of the system should be visible to end-users. This is mostly an operational function and is achieved using an appropriate program interface. General perspective: The developmental state of the system should be represented. This is done by using documents, diagrams, drawings, representative models, prototypes etc. Specifications are used to ensure accurate system representation. Products must still provide system status visibility to end-users, including during production, operation and maintenance.
2	Match between system and the real world	Contextualisation and accuracy	IS perspective: A system should be realistic and linked to a real-world application in order to prevent confusion when making decisions and executing tasks. General perspective: System diagrams should be realistic and accurate, and must be derived from operational analyses (thus creating a link to the real world). Documents and models should be placed in context. Accurate and comprehensive ontologies are required. Systems must be realistically presented during development and products must be aligned with the context of the real world.
3	User control and freedom	(i) Quality and (ii) Requirements management	IS perspective: A user should be able to interact with a program, being allowed the ability to change settings, configurations, input values etc. This allows the user to execture control. General perspective: Subject to limitation, the ability of a user to control requirements equates to quality (i.e. the voice of the customer [114]). Relevant users must be able to change requirements (as with Agile development) utilizing a form of supportive configuration management. Designers should be allowed freedom within constraints. Products must still allow user control.
4	Consistency and standards	Consistency and standards	IS perspective: Both *de facto* and formal industry standards are used for software programmes, resulting in consistency (when a standard IDE is used, for example). General perspective: Standard methods and techniques for representing systems must be used, for example, functional analyses using standard flows and architecture models. Standard documentation methods, development methods and tools should be used. Products should still adhere to standards and be consistent across product families (i.e variants of a product).
5	Error prevention	Technical risk management	IS perspective: Users should be prevented from making mistakes by means of pre-planned coding of interactions. Preventing incorrect user entries and choices is an example. General perspective: Error prevention, in general, requires following defined processes and procedures. Risks must be managed to address potential errors (failures) using standard risk management methods, including techniques such as failure mode analysis, for example. Products must still prevent users from erring by design.
6	Recognition rather than recall	Constructivism / familiarity	IS perspective: Users should be familiar with concepts (screens, sequences, etc.) rather than simply recalling them from memory. General perspective: Constructivism deals with the process of being able to relate to familiar concepts, also when learning [28]. In systems, this implies limited use of foreign concepts, and when doing so, relating those concepts to concepts familiar to users. Products must still employ familiar concepts.
7	Flexibility and efficiency of use	Balance between agility and efficacy	IS perspective: User should, within predetermined constraints, be able to exercise options without excessive effort (having to make too many choices, selections etc.). That is, different ways of achieving a goal should be allowed. General perspective: Users should have options when defining products (clients, designers etc.) within efficacy limits. Resources should support agility as client requirements often change, with options rather than uncontrolled freedom. Products must support options, within limits of efficacy.
8	Aesthetic and minimalist design	Complexity reduction – Occam's razor	IS perspective: Users get confused with overly complex interfaces, hence the requirement for simplicity. Aesthetically pleasing interfaces are also more readily accepted. General perspective: Reliable systems usually have limited system elements. Representations of systems should be simplistic and limited in complexity. System hierarchies should be employed (layers of complexity) with not too much detail on each layer. Products must still be simplistic and minimalistic.
9	Help users recognise, diagnose, and recover from errors	(i) Validation, verification, and (ii) Configuration management	IS perspective: Users should be assisted to recognise errors by appropriate program guidance and feedback. Options for recovery must be present and diagnosis is used to assist users in determining underlying errors. General perspective: (i) During development, verification and validation are used to ensure design errors are identified and addressed. (ii) Change management (configuration management in the broader sense) is used to allow rectification of errors. Root-cause analysis is a standard tool to assist with the diagnosis of design errors. Products must still allow users to recognise and recover from errors.
10	Help and documentation	(i) Support and (ii) Communication	IS perspective: From an interface, it should be possible to access a form of assistance (the familiar F1 help function) with documentation to assist with training, fault-finding and other forms of assistance such as access to experts (support centres, experts etc)c. General perspective: (i) Development assistance must be provided for complex systems in the form of specialists, for example. This not only improves user experience but also product development time (at a cost). Other forms of support are available in the form of access to tools, methods, training etc. (ii) Communication is key to full life cycle usability, with standard documentation, informative design sessions and meetings, use of representative models etc. Products must be supported with documents and other available resources, including online support, expert input etc.

C. Usability heuristics applied to a system full life cycle

In what follows, a basic form of validation is provided to show how generalized heuristics can address human-related issues in different life cycle phases. General issues relating to human factors were identified from studies and listed in matrices. Applicable heuristics were identified that may potentially address usability shortfalls.

1. Requirements analysis:

A diagram for requirements analysis is shown in Fig. 1. Requirements analysis (RA) is the process of eliciting and recording system requirements. RA is a critically important phase in the product development process. The success of phases that follow depends on requirements (specifications) drawn up here. Tools like requirement management (RM) tools should be used to coordinate the RM process and information.

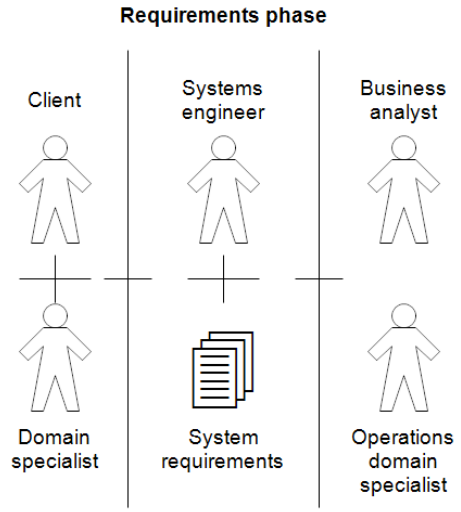

Fig. 1. Users for requirements analysis phase

While RM tools can be very helpful, they often have limitations. Challenges in the RM phase are shown in Table II. The first column presents issues from literature and the second column provides generalised heuristics that give potential solutions to that issue.

TABLE II. REQUIREMENTS ANALYSIS ISSUES AND USABILITY HEURISTICS

Lack of traceability, tracking the effect of changes, data synchronisation and version control [28], [30], [31]	3, 9
Lack of information access and visibility [28]	1, 10(ii)
Lack of context for actions and decisions, differences in client and development team ontologies, unclear problem statements [28], [31], [33]	1, 2, 3, 10(ii)
Mostly text-based (the complexity often requires diagrams as well) [30]	1, 2, 4, 8, 10(ii)
Specifications drawn from invalid premises and assumptions [30]	3, 5, 10(ii)
Lack of non-functional requirements (usability, etc.) [30], [33]	2, 4, 5
Conflicting goals between the client and vendor [29]	3
Lack of input information from clients and managers, low client involvement, sign-off issues [29], [31]–[33]	3, 5
Low or no direct contact between the development team and the client [32]	3, 5 10
Inadequate risk evaluations [33]	5

2. Design, integration, and testing:

A diagram for design, implementation, and testing analysis is shown in Fig. 2. The design, integration, and testing phase includes preliminary design, detail design, prototyping, and testing. It is clear that the number of human interfaces is high and challenges will be as numerous. Essentially, every human interface has the potential for failure due to the complexity of the development effort and should be considered.

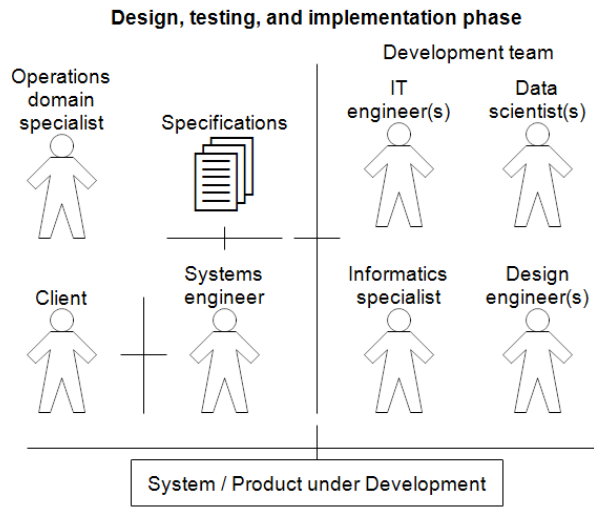

Fig. 2. Users for design, testing, and implementation analysis phase

Using general heuristics to address usability shortfalls with design, integration and testing should resolve challenges as obtained from literature in the design, integration, and testing phase as shown in Table III. The first column contains issues and the second column refers to generalised heuristics that may provide solutions to that issue.

TABLE III. DESIGN, INTEGRATION, AND TESTING ISSUES AND USABILITY HEURISTICS

Inadequate technical risk management [37]. Not only a usability issue, but results in humans not being able to prevent errors.	1, 3, 5, 9
Information synchronisation and standardisation, lost/corrupted data/information, version control, lack of documenting knowledge [36]–[38], [41], [45].	4, 9, 10
Lack of tools or resources (introducing usability issues) [26], [37]	5, 7, 9, 10(i)
Not including specialist consultation where needed, lack in knowledge integration and team approach, one-sided decisions [27], [34], [36]–[38]	1, 3, 10(ii)
Lack of decision integrity and traceability, hiding issues, incorrect assumptions and units [37], [42]	3, 10(ii)
Different ontologies between development team members [27], [37]	1, 2, 3(ii), 10(ii)
Low client involvement, decision changes by client [27], [37]	2, 3, 6, 8, 10(ii)
Lack of design information access and visibility, inadequate requirements [27], [37], [38], [45]	1, 2, 3(ii), 10
Quality control [27], [38]	3, 4, 9(i)

3. Product manufacturing:

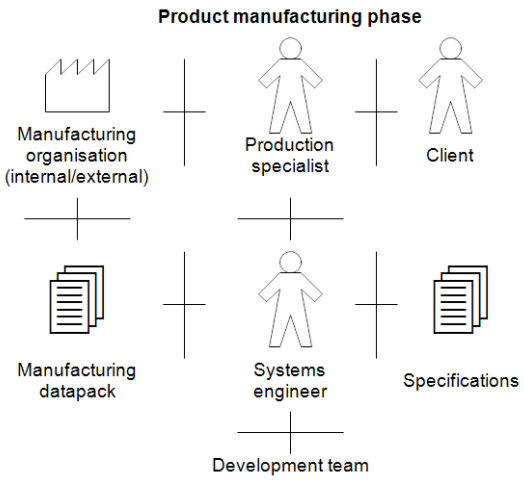

Fig. 3. Users for product manufacturing analysis phase

A diagram for product manufacturing analysis is shown in Fig. 3. Production is an often neglected phase of a system full life cycle as engineers sometimes consider a job completed after development. However, it is not uncommon for large test support systems to be developed after the formal product design, integration and testing have been completed. As a result, usability is often neglected for manufacturing, with production and test personnel suffering the consequences. This may eventually result in a lack of product quality.

TABLE IV. PRODUCT MANUFACTURING ISSUES AND HEURISTICS

Lack of decision support, information synchronisation, issues with data handling, issues with information access and visualisation, transparency, inadequate concurrent engineering [26], [27], [39]	1, 3(ii), 10
Inadequate risk management, behaviour forecasting, predictability [39]	3, 5
Lack of control and real-time control at all levels of the system, automation control, and inadequate quality control [35], [39]	1, 3,
Security and safety aspects issues regarding data management and production processes [39] (also addressed by physical ergonomics, not only heuristics)	5
Communication issues between humans, machines, computers, robots, etc., real and virtual system fusion [35], [39]	1, 2, 10(ii)
Robustness and remote diagnosis at all system levels [39]	1, 9, 10(ii)
Design and design change management issues [26], [39]	3(ii), 9(ii)
Communication issues between manufacturer, supplier, manager and client [26], [35], [39], [41]	10(ii)
Interdepartmental disharmony, ontology differences [35]	1, 2

Production challenges are shown in Table IV. The first column contains usability issues as found from literature and the second column shows generalised heuristics that could provide solutions to issues.

4. Operation and maintenance:

A diagram for operation and maintenance analysis is shown in Fig. 4. Operation and maintenance include installation, day-to-day operation, and preventative and corrective maintenance. Major issues found during operation (and associated maintenance) may lead to design changes, which are costly. An efficient and accurate design upstream will mostly avoid such issues. As product usability is mostly considered in the actual product design, which is not the focus of this paper, attention will be on usability in the operations and maintenance *support* systems.

Issues in the operation and maintenance phase are shown in Table V. The first column is the issues and the second column is the generalised heuristics that provide solutions to a particular issue.

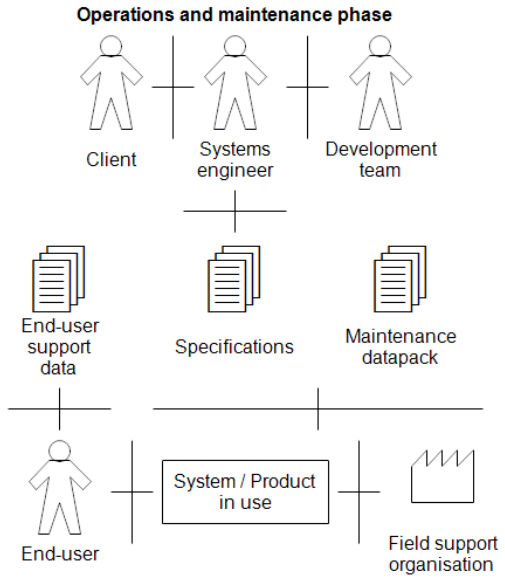

Fig. 4. Users for operation and maintenance analysis phase

TABLE V. OPERATION AND MAINTENANCE ISSUES AND USABILITY HEURISTICS

Lack of designing for tolerance and maintainability [44]	2, 3
Quality control issues [44], [46]	3
Issues with dismantling and task execution [44]	10
Information integration, access and visibility, inadequate data collection, processing and distribution [44], [46]	1, 10(ii)
Inadequate system monitoring, diagnosis, prognosis, lack of preventative maintenance [44]	1, 5, 9(i)
Lack of decision support [44], [46]	10(i)
Inadequate installation, operations, and maintenance guidance for technicians, specialists, end users [44]	10
Lack of resources (including sensors, etc.), time and cost, expensive resources [44], [46]	1, 10
Maintenance adaptivity to change [44]	7

5. System / product retirement:

A diagram for system/product retirement analysis is shown in Fig. 5. A diagram is not provided here due to limited space, but the process essentially implies (a) re-engineering of existing products or systems [57], or (b) recycling of usable components, or (c) destruction.

TABLE VI. RETIREMENT ISSUES AND USABILITY HEURISTICS

Balance between environmental friendliness and client requirements, costs and profits [44]	3
Disassembly issues, including unclear disassembly procedures [44]	10
Difficulty acquiring/using product usage data and systems documentation for re-engineering improvements [40]	9(ii), 10
Difficulty converting environmentally green principles into functional design rules, and including it in design tools [43]	3, 4
Lack of broadly agreed standard environmental impact assessment methods, random external factors [43]	4

Retirement may involve dismantling of a product, removal of a product from the site, disposing and recycling product components. [40]. Issues in the retirement phase are shown in Table VI. The first column contains issues as obtained from literature and the second column contains generalised heuristics that could provide solutions to that specific issue.

V. Reflection

Translation of Nielsen's ten heuristics resulted in an interesting set of generalised heuristics that relate closely to systems engineering process functions [6]. It is not surprising to see the following familiar functions that directly relate to systems engineering:

- System representation (to describe system development status / configuration);

- Quality and requirements management (as a means to ensure client input and control);
- Consistency and standards (in all development efforts, standars play a role to ensure consistency);
- Technical risk management (to prevent design and other errors or failures);
- Verification and validation (as a method to identify design errors);
- Change / configuration management (as a means to recover from errors);
- Specialist support (in the form of relevant specialists and other resources); and
- Communication (not only for assistance, but also to record and convey design and other system data).

In addition to the above, it is instructive to find aspects such as the following:

- Contextualization and accuracy (to ensure system context is preserved and accurate);
- Constructivism and familiarity (to simplify understanding, capture and keep client involvement);
- Balance between agility and efficacy (which is a contentious issue in development); and
- Complexity reduction (using system layers to support understanding in, for example, system representation).

From the above, it is clear that, by following a systems engineering approach with effective contextualization, simplification by means of constructivism, and complexity reduction, and sound communication principles, an opportunity is created for establishing usability for designers. There may be more factors that will be included from future research, but the baseline created here can already support development efforts at this point.

VI. Conclusion

Research was done to understand the notion of usability in a system full life cycle. From a literature study, Nielsen's ten heuristics were selected as representative of IoT system usability factors. A generalised set of heuristics was derived and considered as solutions to issues relating to usability in each life cycle phase as identified from a literature survey. It was found that issues could be addressed by following usability principles and that these generalised heuristics relate closely to system engineering functions.

Physical ergonomics and system safety were not specifically considered as these are well known. The focus was maintained on general usability heuristics, with the knowledge that ergonomics and safety apply to all life cycle phases.

To conclude, it may be stated that systems engineering efforts, when managed properly, can provide a framework that supports usability. In addition, general principles such as contextualization, constructivism, agility and complexity reduction also apply.

It is acknowledged that the study was by no means comprehensive, but the results show that usability in a full life cycle is interesting. Usability allows developers to better perform in a project by using appropriate resources, including tools, methods and techniques.

VII. Authors and Affiliations

Johann EW Holm is an associate professor at the School for Electrical, Electronic and Computer Engineering of the North-West University in South Africa. He is a member of the IEEE and a member of INCOSE, and is actively involved in research and development, as well as consulting in operations and product development.

Esté du Plessis is a postgraduate student at the School for Electrical, Electronic and Computer Engineering of the North-West University in South Africa. She did her undergraduate degree in Computer and Electronic Engineering at the North-West University in South Africa and is currently

busy with her Master's degree in Computer and Electronic Engineering.

7.1.1.1.1 Acknowledgment

The authors would like to acknowledge the North-West University for academic support and for making resources available. Finally, Jericho Systems is acknowledged for their financial support and for allowing access to development projects where usability principles could be verified.

7.1.1.1.2 References

[1] A. Bhattacherjee, *Introduction to Research, Social Science Research: Principles, Methods, and Practices*. 2012.

[2] A. R. Hevner, "A Three Cycle View of Design Science Research," *Scand. J. Inf. Syst.*, vol. 19, no. 2, 2007.

[3] A. Hevner, S. March, J. Park, and S. Ram, "Design Science Research in Information Systems," *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004.

[4] Lindgren, Sein, Henfridsson, Rossi, and Purao, "Action Design Research," *MIS Q.*, vol. 35, no. 1, p. 37, 2017.

[5] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2008.

[6] B. S. Blanchard and W. J. Fabrycky, *Systems Engineering and Analysis*, Third. New Jersey: Prentice Hall, 1998.

[7] T. M. Papp, "CrashAppTM –Concurrent Multiple Stakeholder Evaluation of a DSR Artefact," no. October, p. 136, 2017.

[8] M. T. Mullarkey and A. R. Hevner, "New Horizons in Design Science: Broadening the Research Agenda," *Desrist 2015*, vol. LNCS 9073, pp. 121–134, 2015.

[9] W. Quesenbery, "What Does Usability Mean: Looking Beyond 'Ease of Use' - Whitney Interactive Design," *WQUsability*. pp. 1–8, 2015.

[10] International Organization For Standardization, "ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability." 1998.

[11] I. D. Foundation, "Usability," *Interact. Des. Foundadtiontion*, 2013.

[12] J. Nielsen, "10 Usability Heuristics for User Interface Design," *Nielsen Norman Gr.*, p. 1, 1995.

[13] J. N. R. Molich, "Heuristic Evaluation of User Interfaces," *CHI'90 Proc.*, Apr. 1990.

[14] J. Nielsen, "Enhancing the explanatory power of usability heuristics," *Hum. Factors Comput. Syst.*, pp. 152–158, 1994.

[15] J. Folmer, Eelke; Bosch, "Architecting for Usability; a Survey," *J. Syst. Softw.*, no. 70, pp. 61–78, 2004.

[16] M. Van Welie, G. C. Van Der Veer, and A. Eliëns, "Breaking down Usability," *Proc. INTERACT*, pp. 613–620, 1999.

[17] O. L. Huye, "University's official website design framework using human computer interaction rules," Universiti Malaysia Pahang.

[18] ISO/IEC, "ISO 9126: Software Engineering - Software Product Quality - Part 1: Quality Model." 2001.

[19] D. Zimmermann, "Universal Acess in Human Computer Interaction. Coping with Diversity," vol. 4554, no. July 2007, 2007.

[20] S. Winter, S. Wagner, and F. Deissenboeck, "Engineering Interactive Systems 2008," *Eng. Interact. Syst. 2008*, pp. 106–122, 2008.

[21] J. Nielsen, "Usability 101: Introduction to Usability," *Nielsen Norman Gr.*, 2012.

[22] E. Kiliç Delice and Z. Güngör, "The usability analysis with heuristic evaluation and analytic hierarchy process," *Int. J. Ind. Ergon.*, vol. 39, no. 6, pp. 934–939, 2009.

[23] W. Quesenbery, "What Does Usability Mean: Looking Beyond 'Ease of Use,'" *WQUsability*, pp. 1–8, 2014.

[24] B. Caldwell, C. Michael, L. G. Reid, and G. Vanderheiden, "Web Content Accessibility Guidelines (WCAG) 2.0," *W3C*, 2008. [Online]. Available: https://www.w3.org/TR/2008/REC-WCAG20-20081211/#intro-layers-guidance.

[25] D. J. Mayhew, *The Usability Engineering Lifecycle: a practitioner's handbook for user interface design*. Morgan Kaufmann Publishers, 1999.

[26] O. Isaksson, T. C. Larsson, and A. Ö. Rönnbäck, "Development of product-service systems: challenges and opportunities for the manufacturing firm," *J. Eng. Des.*, vol. 20, no. 4, pp. 329–348, Aug. 2009.

[27] L. Ballard, G.; Koskela, "On the agenda of design management research," in *Proceedings of the International Conference on Lean Construction*, Guaruja, Brazil, 1998.

[28] J. Kelanti, M., Hyysalo, P. Kuvaja, M. Oivo, and A. Välimäki, "A Case Study of Requirements Management: Toward Transparency in Requirements Management Tools," in *Proceedings of the Eighth International Conference on Software Engineering Advances (ICSEA 2013)*, 2013, pp. 597–604.

[29] J. M. Bhat, M. Gupta, and S. N. Murthy, "Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing," *IEEE Softw.*, vol. 23, no. 5, pp. 38–44, Sep. 2006.

[30] M. Weber and J. Weisbrod, "Requirements engineering in automotive development-experiences and challenges," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2003, pp.

331–340.

[31] J. A. Bubenko, “Challenges in requirements engineering,” in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, 1995, pp. 160–162.

[32] L. Lehtola, M. Kauppinen, and S. Kujala, “Requirements Prioritization Challenges in Practice,” no. May 2014, 2004, pp. 497–508.

[33] B. Boehm and R. Turner, “Management Challenges to Implementing Agile Processes in Traditional Development Organizations,” *IEEE Softw.*, vol. 22, no. 5, pp. 30–39, Sep. 2005.

[34] R. J. Chapman, “The controlling influences on effective risk identification and assessment for construction design management,” *Int. J. Proj. Manag.*, vol. 19, no. 3, pp. 147–160, Apr. 2001.

[35] A. Griffin and J. R. Hauser, “Patterns of Communication Among Marketing, Engineering and Manufacturing—A Comparison Between Two New Product Teams,” *Manage. Sci.*, vol. 38, no. 3, pp. 360–373, Mar. 1992.

[36] P. Tzortzopoulos and C. Formoso, “Considerations on application of lean construction principles to design management,” *Proc. IGLC-7*, pp. 335–344, 1999.

[37] R. Owen *et al.*, “Challenges for integrated design and delivery solutions,” *Archit. Eng. Des. Manag.*, vol. 6, no. SPECIAL ISSUE, pp. 232–240, 2010.

[38] L. Koskela, P. Huovila, and J. Leinonen, “Design management in building construction: From theory to practice,” *J. Constr. Res.*, vol. 03, no. 01, pp. 1–16, Mar. 2002.

[39] L. Monostori, “Cyber-physical Production Systems: Roots, Expectations and R&D Challenges,” *Procedia CIRP*, vol. 17, pp. 9–13, 2014.

[40] F. Tao, Y. Wang, Y. Zuo, H. Yang, and M. Zhang, “Internet of Things in product life-cycle energy management,” *J. Ind. Inf. Integr.*, vol. 1, pp. 26–39, Mar. 2016.

[41] V. A. Mabert and M. A. Venkataramanan, “Special Research Focus on Supply Chain Linkages: Challenges for Design and Management in the 21st Century,” *Decis. Sci.*, vol. 29, no. 3, pp. 537–552, Jul. 1998.

[42] J. C. Knight, “Safety critical systems,” in *Proceedings of the 24th international conference on Software engineering - ICSE '02*, 2002, p. 547.

[43] W. J. Glantschnig, “Green design: an introduction to issues and challenges,” *IEEE Trans. Components, Packag. Manuf. Technol. Part A*, vol. 17, no. 4, pp. 508–513, 1994.

[44] S. Takata *et al.*, “Maintenance: Changing Role in Life Cycle Management,” *CIRP Ann.*, vol. 53, no. 2, pp. 643–655, 2004.

[45] A. Meissner, T. Luckenbach, T. Risse, T. Kirste, and H. Kirchner, “Design Challenges for an Integrated Disaster Management Communication and Information System,” *First IEEE Work. Disaster Recover. Networks*, no. DIREN 2002, 2002.

[46] J. P. Liyanage and U. Kumar, “Towards a value-based view on operations and maintenance performance management,” *J. Qual. Maint. Eng.*, vol. 9, no. 4, pp. 333–350, Dec. 2003.

[47] K. L. Chin, J.P.; Diehl, V.A.; Norman, “Development of an instrument measuring user satisfaction of the human-computer interface,” *Proc. CHI'88 Conf. Hum. Factors Comput. Syst.*, pp. 213–218, 1988.

[48] F. D. Davis, “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology,” *MIS Q.*, vol. 13, no. 3, p. 319, 1989.

[49] R. J. Lewis, “IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. Boca Raton, FL: Human Factors Group,” *IBM Tech. Rep.*, vol. 54, no. 1, p. 786, 1993.

[50] J. Dul *et al.*, “A strategy for human factors/ergonomics: Developing the discipline and profession,” *Ergonomics*, vol. 55, no. 4, pp. 377–395, 2012.

[51] INCOSE, “Systems Engineering,” *Systems Engineering*, 2010. [Online]. Available: https://www.incose.org/systems-engineering.

[52] M. C. Davis, R. Challenger, D. N. W. Jayewardene, and C. W. Clegg, “Advancing socio-technical systems thinking: A call for bravery,” *Appl. Ergon.*, vol. 45, no. 2 Part A, pp. 171–180, 2014.

[53] P. Carayon, “Human factors of complex sociotechnical systems,” *Appl. Ergon.*, vol. 37, no. 4 SPEC. ISS., pp. 525–535, 2006.

[54] J. R. Wilson, “Fundamentals of systems ergonomics/human factors,” *Appl. Ergon.*, vol. 45, no. 1, pp. 5–13, 2014.

[55] A. Dillon, “Group dynamics meet cognition : applying socio- technical concepts in the design of information systems,” *New SocioTech Graffiti Long Wall*, pp. 119–126, 2000.

[56] R. Jiang, *Introduction to Quality and Reliability Engineering*, XXII. Berlin: Springer, 2015.

[57] R. G. Du Preez, “Deployment, re-engineering and risk analysis of a C-band weather radar to build local capacity in South Africa,” North-West University, 2017.

[58] M. Hermann, T. Pentek, and B. Otto, “Design principles for industrie 4.0 scenarios,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2016-March, pp. 3928–3937, 2016.

[59] A. Gilchrist, *Industry 4.0: The Industrial Internet of Things*. New York: Springer Science+Business Media, 2016.

[60] V. Alcácer and V. Cruz-Machado, “Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems,” *Eng. Sci. Technol. an Int. J.*, no. xxxx, 2019.

[61] E. A. Lee, “Cyber physical systems: Design

challenges," *Proc. - 11th IEEE Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput. ISORC 2008*, pp. 363–369, 2008.

[62] A. Radziwon, A. Bilberg, M. Bogers, and E. S. Madsen, "The smart factory: Exploring adaptive and flexible manufacturing solutions," *Procedia Engineering*, vol. 69. pp. 1184–1190, 2014.

[63] S.-H. Yang, *Wireless Sensor Networks*. London: Springer London, 2014.

[64] G. Koschnick, M. Hankel, and B. Rexroth, "RAMI 4.0-Structure The Reference Architectural Model Industrie 4.0 (RAMI 4.0) Contact: Reference Architectural Model Industrie 4.0," vol. 0, no. April, 2015.

[65] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0 – An Introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[66] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[67] N. K. Mukhopadhyay, S.C.; Suryadevara, *Internet of Things*, 9th ed., vol. 9. Cham: Springer International Publishing, 2014.

[68] D. Kiritsis, "Closed-loop PLM for intelligent products in the era of the Internet of things," *CAD Comput. Aided Des.*, vol. 43, no. 5, pp. 479–501, 2011.

[69] H. C. Y. Chan, "Internet of Things Business Models Internet of Things (IoT), Business Model, Strategy and Tactic in IoT, Case Studies in IoT," *J. Serv. Sci. Manag.*, vol. 8, no. 4, pp. 552–568, 2015.

[70] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization," *Comput. Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[71] J. Peschke, A. Luder, and H. Kuhnle, "The PABADIS'PROMISE architecture - a new approach for flexible manufacturing systems," *IEEE*, vol. 1, pp. 491–496, 2006.

[72] IEA, *Definition and Domains of Ergonomics*. .

[73] J. Dul and W. P. Neumann, "Ergonomics contributions to company strategies," *Appl. Ergon.*, vol. 40, no. 4, pp. 745–752, 2009.

[74] B. M. Kleiner, "Macroergonomics: Analysis and design of work systems," *Appl. Ergon.*, vol. 37, no. 1 SPEC. ISS., pp. 81–89, 2006.

[75] B. T. Karsh, P. Waterson, and R. J. Holden, "Crossing levels in systems ergonomics: A framework to support 'mesoergonomic' inquiry," *Appl. Ergon.*, vol. 45, no. 1, pp. 45–54, 2014.

[76] H. Haines, J. R. Wilson, P. Vink, and E. Koningsveld, "Validating a framework for participatory ergonomics (the PEF)," *Ergonomics*, vol. 45, no. 4, pp. 309–327, 2002.

[77] IBM Corporation, "Common User Access (CUA) guidelines," *IBM Knowledge Center*, 2014. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.f54dg00/cuahlp.htm.

[78] S. J. Dorey, "Common User Access -- Principles of GUI Design," *Princ. User Interface Des.*, pp. 1–11, 2007.

[79] Microsoft, "User Interface Principles - Windows applications _ Microsoft Docs," *Windows Dev Center*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/desktop/appuistart/-user-interface-principles. [Accessed: 14-Apr-2019].

[80] ISO, "ISO 9241-14:1997 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 14: Menu dialogues," 1997. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-14:ed-1:v1:en.

[81] ISO/IEC, "ISO 11581-3, Information technology — User system interfaces and symbols — Icon symbols and functions — Part 3: Pointer icons," 2000. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en.

[82] ISO/IEC, "ISO_IEC 11581-10_2010 - Information technology -- User interface icons -- Part 10_ Framework and general guidance," 2010. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en.

[83] I. Apple Computer, *Macintosh Human Interface Guidelines*. Addison-Wesley Publishing Company, 1995.

[84] Interaction Design Foundation, "What is User Interface (UI) Design? | Interaction Design Foundation." p. 1, 2018.

[85] K. Pernice, "F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant (Even on Mobile)," *NNGroup*, 2017. [Online]. Available: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/. [Accessed: 15-Apr-2019].

[86] KDE Human Interface Guidelines, "KDE Human Interface Guidelines — Human Interface Guidelines documentation," *KDE Human Interface Guidelines*. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:11581:-3:ed-1:v1:en. [Accessed: 15-Apr-2019].

[87] A. Phocaides, "Ch 10: The center pivot irrigation system," in *Pressurized Irrigation Techniques*, Second., Rome: FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, 2007, pp. clix–clxxvi.

[88] Senter360, "Senter360," *Senter360*, 2017. [Online]. Available: http://www.senter360.co.za/. [Accessed: 27-Feb-2019].

[89] Global Beef, "Sustainable, Profitable Agriculture." .

[90] Irrigation.Education Community, "How a Center Pivot Irrigation Machine Works," *irrigation.education*,

2016. [Online]. Available: http://blog.irrigation.education/blog/how-a-center-pivot-works. [Accessed: 09-Jul-2019].

[91] X. Dong, M. C. Vuran, and S. Irmak, "Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems," *Ad Hoc Networks*, vol. 11, no. 7, pp. 1975–1987, 2013.

[92] C. R. Camp, E. J. Sadler, D. E. Evans, L. J. Usrey, and M. Omary, "Modified Center Pivot System for Precision Management of Water and Nutrients," *Appl. Eng. Agric.*, vol. 14, no. 1, pp. 23–31, 1998.

[93] Valley, "ICON10." pp. 1–2, 2018.

[94] Valley, "ICON5." pp. 1–2, 2018.

[95] Valley, "ICON1." pp. 1–2, 2018.

[96] Valley, "ICONX." pp. 1–2, 2018.

[97] Talbert, "Talbert RAIN User & Installer Manual." pp. 1–24.

[98] Agrico, "Advanced rain user guide." Agrico, pp. 1–9.

[99] Agrico, "Agrico spilpuntbeheerstelle." Agrico, p. 100.

[100] Zimmatic, "Irrigation control field management products." Lindsay Cooperation, Omaha, pp. 1–4, 2017.

[101] Linsay Cooperation, "FIELDNET Irrigated remote irrigation management." Lindsay Cooperation, Omaha, pp. 1–10, 2018.

[102] Lindsay Cooperation, "FIELDNET Pivot Control & FIELDNET Pivot Control Lite." Lindsay Cooperation, Omaha, pp. 1–4, 2018.

[103] Automation Products, "Main Panel One Tower 110V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots110-main-panel-one-tower-110v.html. [Accessed: 27-Feb-2019].

[104] Automation Products, "'Main Panel One Tower 220V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots220--main-panel-one-tower-220v.html. [Accessed: 27-Feb-2019].

[105] Automation Products, "Main Panel One Tower 24V DC," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots24-main-panel-one-tower-24v-dc.html. [Accessed: 27-Feb-2019].

[106] Automation Products, "Main Panel One Tower Inverter 220V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-poti-main-panel-one-tower-inverter-220v.html. [Accessed: 27-Feb-2019].

[107] Automation Products, "Main Panel Two Tower 110V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-ptt110-main-panel-two-tower-110v.html. [Accessed: 27-Feb-2019].

[108] Automation Products, "Main Panel Two Tower 220V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pots220--main-panel-one-tower-220v.html. [Accessed: 27-Feb-2019].

[109] Automation Products, "Main Panel Large 110V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl110-main-panel-large-110v.html. [Accessed: 27-Feb-2019].

[110] Automation Products, "Main Panel Large 220V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl220-main-panel-large-220v.html. [Accessed: 27-Feb-2019].

[111] Automation Products, "Main Panel Large Inverter 220V," *Automation Products*. [Online]. Available: https://www.autoproducts.co.za/product/m-pl220it-main-panel-large-inverter-220v.html. [Accessed: 27-Feb-2019].

[112] M. Abdel-Basset, G. Manogaran, and M. Mohamed, "Internet of Things (IoT) and its impact on supply chain: A framework for building smart, secure and efficient systems," *Futur. Gener. Comput. Syst.*, vol. 86, pp. 614–628, 2018.

[113] J. E. W. Holm, "IIoT - Systems Approach." Potchefstroom, South Africa, 2018.

[114] F. N. Akhras and J. A. Self, "System Intelligence in Constructivist Learning," *Int. J. Artif. Intell. Educ.*, vol. 11, pp. 344–376, 2000.

[115] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE)," *INCOSE MBSE Initiat.*, 2008.

[116] A. Baker, "Simplicity," in *Stanford Encyclopedia of Philosophy*, Winter 201., E. N. . Zalta, Ed. Metaphysics Research Lab, Stanford University, 2016.

[117] F. Nayebi, J. M. Desharnais, and A. Abran, "An expert-based framework for evaluating iOS application usability," *Proc. - Jt. Conf. 23rd Int. Work. Softw. Meas. 8th Int. Conf. Softw. Process Prod. Meas. IWSM-MENSURA 2013*, no. January 2014, pp. 147–155, 2013.

[118] J. E. W. Holm and E. Du Plessis, "Usability – a Full Life Cycle Perspective," *IEEE Africon*, 2019.

[119] D. A. Viljoen, "Synthesis and evaluation of engineering processes for the development of airborne electronic equipment," North-West University, Potchefstroom Campus, 2016.

[120] I. Yüksel, "Developing a Multi-Criteria Decision Making Model for PESTEL Analysis," *Int. J. Bus. Manag.*, vol. 7, no. 24, 2012.