



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

**JONI MINKKINEN**  
**DEVELOPING A REMOTE DIAGNOSTICS APPLICATION**  
**FOR REMOTE HANDLING SYSTEMS OF ITER**

Master of Science thesis

Examiner: Prof. Jouni Mattila  
Examiner and topic approved by the  
Faculty Council of the Faculty of  
Engineering Sciences  
on 29th March 2017

## ABSTRACT

**JONI MINKKINEN:** Developing a Remote Diagnostics Application for Remote Handling Systems of ITER

Tampere University of Technology

Master of Science thesis, 49 pages, 1 appendix page

September 2017

Master's Degree Programme in Automation Engineering

Major: Machine Automation

Examiner: Prof. Jouni Mattila

Keywords: ITER, condition monitoring, remote diagnostics, LabVIEW, application

ITER is an experimental thermonuclear reactor, which is supposed to solve challenges with energy resources in the future. Once the reactor is finished, it will require maintenance operations at times. Because of radiation in the reactor, maintenance operations are remotely controlled and therefore a remote diagnostics application is required for monitoring this remotely controlled maintenance system.

This thesis focuses on developing the Remote Diagnostics Application for remote handling systems of ITER. The application will be developed in co-operation with VTT Technical Research Centre of Finland LTD and Tampere University of Technology. Aim of the project is to develop application for monitoring status and condition of the maintenance equipment and save acquired data into archives for investigation purposes. \*

One of the main requirements for the application is that the application should be customisable and flexible. Application should be usable also with other types of equipment, not only with maintenance equipment. The application is rule-based and the operator should be able to create and edit these rules without aborting or recompiling the application. This caused some architectural problems to solve. The application is developed with graphical LabVIEW development software, which allowed many different approaches to the architectural problem.

As a result of this thesis, a prototype of the Remote Diagnostics Application was created. The prototype was demonstrated for the customer in May 2017. Developing the application will continue and plans for the future development process are already done.

\*The work leading to this thesis has been funded by Fusion for Energy, TEKES, TUT and VTT under Grant F4E-GRT-0689. This thesis reflects the views only of

the author, and Fusion for Energy, TEKES, TUT or VTT cannot be held responsible for any use which may be made of the information contained therein.

# TIIVISTELMÄ

**JONI MINKKINEN:** Etädiagnostiikkaohjelmiston kehittäminen ITERin etähallintajärjestelmiä varten

Tampereen teknillinen yliopisto

Diplomityö, 49 sivua, 1 liitesivu

Syyskuu 2017

Automaatiotekniikan diplomi-insinöörin koulutusohjelma

Pääaine: Hydraulikka ja automatiikka, koneautomaatio

Tarkastaja: Prof. Jouni Mattila

Avainsanat: ITER, kunnonvalvonta, diagnostiikka, LabVIEW, ohjelmisto

ITER on kokeellinen fuusioreaktori, jonka on tarkoitus ratkaista energiantuotantoon liittyvät haasteet tulevaisuudessa. Reaktorin valmistuttua se tulee tarvitsemaan ajoittain myös huoltoa. Reaktorissa syntyvän säteilyn vuoksi huoltotoimenpiteet suoritetaan etäoperoidusti, ja tämän takia tarvitaan myös etädiagnostiikkaa etäoperoidun huoltolaitteiston tilan seurantaan varten.

Tämä opinnäytetyö keskittyy etädiagnostiikkaohjelmiston kehitykseen. Ohjelmisto kehitettiin yhteistyössä Teknologian tutkimuskeskus VTT Oy:n sekä Tampereen teknillisen yliopiston kanssa. Projektin tavoitteena oli luoda ohjelmisto, jolla on mahdollista tarkkailla huoltojärjestelmän tilaa ja kuntoa, sekä tallentaa järjestelmästä saatavaa dataa myöhempää analysointitarvetta varten. \*

Yksi tärkeimmistä vaatimuksista ohjelmistolle oli sen muokattavuus ja joustavuus. Ohjelmisto ei tule olemaan pelkästään huoltolaitteiston, vaan myös yleisemmin muiden laitteistojen käyttöön soveltuva. Ohjelmiston tuli olla tarpeiden mukaan muokattava, jotta sitä voidaan käyttää erilaisten laitteiden seurantaan ja tarkkailuun. Ohjelmisto on sääntöpohjainen ja käyttäjän tulee pystyä luomaan ja muokkaamaan sääntöjä aina tarvittaessa ilman ohjelmiston sulkemista tai uudelleenkäntämistä. Tämä aiheutti haasteita ohjelmiston arkkitehtuurin kanssa. Ohjelmiston kehittämiseen käytettävä graafinen LabVIEW-kehitysympäristö mahdollisti kuitenkin helposti erilaisten ratkaisujen toteuttamisen.

Työn tuloksena syntyi etädiagnostiikkaohjelmiston prototyyppi, jota myös esiteltiin asiakkaalle toukokuussa 2017. Ohjelmiston kehitystä on tarkoitus jatkaa ja suunnitelmia sitä varten on jo tehty.

\*The work leading to this thesis has been funded by Fusion for Energy, TEKES, TUT and VTT under Grant F4E-GRT-0689. This thesis reflects the views only of



the author, and Fusion for Energy, TEKES, TUT or VTT cannot be held responsible for any use which may be made of the information contained therein.

## PREFACE

The process to write this Master of Science thesis started in the beginning of the year 2017. The process was finished during September in the same year. This thesis was carried out in Laboratory of Automation and Hydraulics at Tampere University of Technology.

I would like to thank Professor Jouni Mattila for the possibility to work at Laboratory of Automation and Hydraulics. Also thanks to Senior Scientist Hannu Saarinen from VTT, who helped me with the LabVIEW development platform and gave me many good advices for the application. Also many thanks for Senior Scientist Jarmo Alanen from VTT, who was giving instructions for developing the Remote Diagnostics Application and gave me advices for writing this thesis.

Tampere, 20.9.2017

Joni Minkkinen

# TABLE OF CONTENTS

1. Introduction . . . . .	1
1.1 ITER and Remote Diagnostics Application . . . . .	1
1.2 Objective and Structure of the Thesis . . . . .	2
2. Condition Monitoring and Diagnostics . . . . .	4
2.1 Terminology of Condition Monitoring . . . . .	4
2.1.1 Condition Monitoring . . . . .	4
2.1.2 Dependability . . . . .	5
2.1.3 Durability, and Safety and Security . . . . .	5
2.1.4 Availability . . . . .	6
2.1.5 Reliability . . . . .	6
2.1.6 Recoverability . . . . .	6
2.1.7 Maintainability . . . . .	7
2.1.8 Maintenance Support Performance . . . . .	7
2.2 Diagnostics and Prognostics . . . . .	7
2.3 Purposes of Condition Monitoring and Diagnostics . . . . .	8
2.4 Condition Monitoring and Diagnostics Methods . . . . .	8
2.4.1 Artificial Intelligence Based Diagnostics Method . . . . .	8
2.4.2 Model Based Diagnostics Method . . . . .	9
2.5 Maintenance Strategies . . . . .	10
2.5.1 Breakdown Maintenance Strategy . . . . .	11
2.5.2 Preventative Maintenance Strategy . . . . .	12
2.5.3 Condition-Based Maintenance Strategy . . . . .	13
2.6 Suitable Maintenance Strategies for Remote Diagnostics Application .	15
3. Development of the Remote Diagnostics Application . . . . .	17
3.1 Remote Handling Systems . . . . .	17
3.2 The Reason for Developing a New Remote Diagnostics Application .	18
3.3 Development Process of the Remote Diagnostics Application . . . . .	19

3.3.1	Waterfall Model . . . . .	20
3.3.2	Spiral Model . . . . .	20
3.3.3	Models Used with Development Process of the Remote Diagnostics Application . . . . .	22
3.4	Development Platform . . . . .	22
3.4.1	Development with LabVIEW . . . . .	23
3.4.2	Alternative Development Platforms . . . . .	25
3.5	Requirements for the Remote Diagnostics Application . . . . .	25
3.6	ISO 13374 Based Guidelines for Development Process . . . . .	26
3.6.1	Condition Assessment Blocks . . . . .	27
3.6.2	Monitoring Features . . . . .	28
4.	Implementation of the Remote Diagnostics Application . . . . .	31
4.1	Three-Layered Architecture . . . . .	31
4.1.1	Sub Panels . . . . .	33
4.1.2	Dynamic Calls . . . . .	34
4.2	Workbenches . . . . .	35
4.3	Diagnostics Primitives . . . . .	36
4.4	Status Monitor . . . . .	36
4.5	Event Monitor . . . . .	38
5.	Analysis of the Remote Diagnostics Application Prototype . . . . .	40
5.1	Current State (as of May 2017) . . . . .	40
5.1.1	Demonstration of the Prototype . . . . .	41
5.2	The Prototype Compared to the ISO 13374 Standard . . . . .	41
5.3	Future Plans . . . . .	42
5.3.1	Test campaign . . . . .	42
5.3.2	GUI Look-and-Feel Improvements and Web Browser Interface . . . . .	43
5.3.3	More Automatic Application . . . . .	44
6.	Conclusions of the Project . . . . .	46
	References . . . . .	48



## LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CBM	Condition Based Maintenance
DTP2	Divertor Test Platform 2
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
IEV	International Electrotechnical Vocabulary
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
RDA	Remote Diagnostics Application
VI	Virtual Instrument
VR	Virtual Reality
VTT	VTT Technical Research Centre of Finland LTD
TUT	Tampere University of Technology

# 1. INTRODUCTION

This introduction chapter will shortly introduce the topic of this Master of Science thesis. Objective and structure of the thesis are introduced in the second section of this chapter.

## 1.1 ITER and Remote Diagnostics Application

ITER is an engineering megaproject. It is aimed to demonstrate fusion as a viable energy source in the future. The name means ‘The Way’ in Latin. Originally it was an acronym for ‘International Thermonuclear Experimental Reactor’ which is no longer used due to the negative connotation of the word ‘thermonuclear’ [15]. The project started nearly 30 years ago and first plasma should be achieved in 2025. The reactor is planned to be operational in 2035. [7]

Because of the massive size of the project, it has been divided between several countries and companies. China, the European Union, India, Japan, Korea, Russia and the United States are taking part to this project. [7] Finland as a part of the European Union has its own part in this project. For example, VTT Technical Research Centre of Finland LTD (VTT) is developing and doing research about remote handling systems for maintenance operations [9].

Due to fusion reaction in ITER, there will be radiation in the reactor [7], which should be avoided by operators because of its harmful effects. Maintenance operations of the reactor has to be done despite the radiation. For this requirement, remotely controlled maintenance systems are developed. With remotely controlled system, maintenance operations are harmless for the operators.

Components of the maintenance equipment should be possible to be monitored all the time. For this requirement, Remote Diagnostics Application (RDA) will be developed by VTT and Tampere University of Technology (TUT). With the RDA, all the components and actuators can be monitored in real time. Purpose of this application is to detect anomalies, problems and faults in real time and beforehand. This provides possibility to prevent unexpected failures and to schedule maintenance

operations. Unexpected shutdowns may become very expensive and take a lot of time. Avoiding them is one of the purposes of this application.

With the RDA, it is also possible to observe history data of the components and actuators. The RDA saves all the data coming from different devices and allows analysing the measurement data afterwards. This makes possible to see slow changes in conditions of the components and actuators. One example of this kind of change is an increasing hysteresis, which may be difficult to see from a raw data coming directly from the faulty device.

The new thing in developing this application is that it will be very dynamic application. It should be possible to add new rules and components without the need of recompiling or restarting the application. In other words, there will be only main application, which will get new features over the time depending on customer requirements. This may be a challenge in terms of developing this application. The application is developed with graphical LabVIEW development environment, which is a product of National Instruments.

There are ready products on the market for fault diagnostics purposes. The problem is that most of these products are aimed for mass produced items so they are not compatible with ITER and does not meet the requirements of flexibility and customisability [12]. By creating a new application for diagnostic purposes, it is possible to create something what is really needed without compromises.

## 1.2 Objective and Structure of the Thesis

This thesis focuses on development process of the RDA. The application will be very complex and will have many different components. Because of that, this thesis is mainly focused on the RDA front end implementation, which is primary task relating to this thesis. Some of the other parts are also discussed in this thesis but with lower level of details.

After this introduction chapter, basic condition monitoring information and terminology is included in the second chapter. Understanding condition monitoring basics is important to fully understand the principles of the remote diagnostics.

Remote diagnostics and purpose of the diagnostic tools are introduced in the third chapter. It is also discussed what remote diagnostics means to ITER and why the new application is required. The development process, development platform and requirements for the application are part of this chapter. There is also ISO 13374



standard available about condition monitoring, and the standard is also introduced in this chapter.

The fourth chapter will concentrate on the front end implementation of the RDA. In this chapter, the architecture of the front end implementation is reviewed. In addition, the idea of the workbenches and primitives is described and different indicators for event monitoring are introduced.

The fifth chapter will concentrate on the results. The prototype version of the application is reviewed. Did the prototype version of the application meet the customer requirements and does it perform as expected? The future plans of the application development will also be handled in this chapter. In the last sixth chapter the conclusions of the project are provided.

All information and figures of the RDA in this thesis are based on the prototype version. The prototype version was released in May 17th 2017. Features added to the application after this date and before thesis' release day are not included in this thesis.

This thesis does not include information about ITER as a fusion reactor. All information about ITER is RDA related. ITER is represented in other theses many times and due to this, it is not relevant to include basic information once again. In order to learn more about ITER fusion reactor, updated information and facts can be found from official ITER website (<https://www.iter.org/> [accessed 9.9.2017]) and Fusion for Energy website (<http://fusionforenergy.europa.eu/> [accessed 9.9.2017]).

## 2. CONDITION MONITORING AND DIAGNOSTICS

Most of the modern systems are equipped with condition monitoring and diagnostics in order to make maintenance operations more effective. Condition monitoring and diagnostics systems can provide a clear picture of the status and the health of the monitored system for the operator. [8, p. 25-8] In this chapter, condition monitoring and diagnostics basics, terminology and different condition monitoring strategies and methods are introduced.

### 2.1 Terminology of Condition Monitoring

International Electrotechnical Commission (IEC) has defined a lot of terminology, which can be used with condition monitoring. IEC has created Electropedia (<http://www.electropedia.org/> [accessed 13.7.2017]), which includes many different terms and definitions. The Electropedia is also known as International Electrotechnical Vocabulary (IEV). Every term in the IEV has a reference number, which makes terms easier to be found. Relevant terminology for this project is defined in Electropedia under subject area 192, which is named as ‘dependability’. Most of the terms used in this section of thesis are found from Electropedia section ‘192-01: basic concepts’. [5]

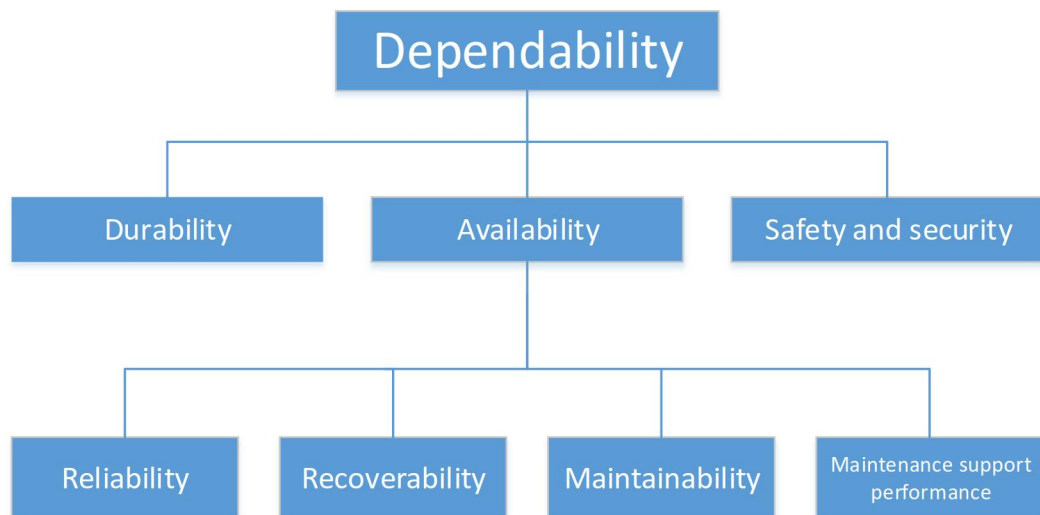
#### 2.1.1 Condition Monitoring

Condition monitoring is one of the most important terms in this project. It is defined as ‘obtaining information about physical state or operational parameters’. Condition monitoring is used to monitor a status of a system. Monitoring operations can be performed manually or automatically with computers. When performing manually, the operator acquires the data with handheld units from monitoring points at planned intervals. When computers are used for monitoring purposes, the computers acquire the data automatically and inform the operator if there are faults in

the system. [5, IEV ref. 192-06-28] [18, p. 13-14] With condition monitoring, the current status of the system is known, but predictions or suggestions of repair for detected faults can not be made; diagnostics is required for this.

### 2.1.2 Dependability

For condition monitoring, the top-level term is dependability. It is defined by IEC as ‘ability to perform as and when required’. This term includes the following three sub-terms: durability, availability, and safety and security. Sometimes durability, and safety and security are not included in the list. [5, IEV ref. 192-01-22] Dependencies of these and sub-terms of availability are presented in Figure 2.1.



**Figure 2.1** Terminology used in condition monitoring. The terms are defined by IEC. Top-level term is called as dependability, which consists of durability, availability, and safety and security. Availability consists of reliability, recoverability, maintainability and maintenance support performance. Durability, and safety and security are not as commonly used as availability. [5, IEV ref. 192-01-22]

### 2.1.3 Durability, and Safety and Security

Durability is defined as ‘ability to perform as required, under given conditions of use and maintenance, until the end of useful life’. Safety and security does not have definition made by IEC. These terms are not as commonly used as availability. [5, IEV ref. 192-01-21, 192-01-22]

Avižienis et al. have split the term safety and security into two terms and defined security as ‘absence of catastrophic consequences on the user(s) and the environment’. In addition, security is included in the top-level term dependability and safety is a sub-term under dependability. [3, p. 13-14] For safety, there is a definition in IEC under subject area ‘Control technology’, and it is defined as ‘freedom from unacceptable risk to the outside from the functional and physical units considered’ [5, IEC ref. 351-57-05].

#### **2.1.4 Availability**

Availability is the most important term after dependability, because it consists of four important sub-terms used in condition monitoring. ‘Availability depends upon the combined characteristics of the reliability, recoverability, and maintainability of the item, and the maintenance support performance’. Availability is defined as ‘ability to be in a state to perform as required’. [5, IEC ref. 192-01-23]

Each sub-term has an impact to availability. As an example, in Chapter 2.5 is a Figure 2.2, which illustrates how availability is affected by amount of maintenance operations and total maintenance costs. In section 2.5, it is described what level of availability is optimal to reach. Optimal availability level depends on application, and therefore high availability is not always the best option [18, p. 4-6].

#### **2.1.5 Reliability**

Reliability is defined by IEC as ‘ability to perform as required, without failure, for a given time interval, under given conditions’. According to IEC, the given time interval should always be clearly stated. It can be for example calendar time or operating cycles. [5, IEC ref. 192-01-24] A passenger car is a good example; high reliability of the car means that it will run to destination without faults. A car with low reliability may stop running before destination. Travel time to the destination is given time interval in this example.

#### **2.1.6 Recoverability**

Recoverability is defined as ‘ability to recover from a failure, without corrective maintenance’. Recovering may require external actions, for example actions from operator. Recovering can be performed also as self-recovering, where no actions is required from operator. [5, IEC ref. 192-01-25]

A good example of this is a computer. Sometimes application may become unresponsive, and the application is possible to be recovered by rebooting the system. This requires external actions but no corrective maintenance. Some applications have self-recovering abilities, when no actions are required.

### **2.1.7 Maintainability**

Maintainability is defined as ‘ability to be retained in, or restored to a state to perform as required, under given conditions of use and maintenance’. Given conditions can be for example location for maintenance operations and maintenance resources. [5, IEC ref. 192-01-27] As an example, mobile devices have usually bad maintainability, because it is not possible to perform any maintenance without special tools. A car has better maintainability, because user can change tyres or running lights easily with basic tools.

### **2.1.8 Maintenance Support Performance**

Maintenance support performance is defined as ‘effectiveness of an organization in respect of maintenance support’ [5, IEC ref. 192-01-29]. In other words, if the organisation has resources for maintenance operations and required know-how available, maintenance support performance is on high level. Poor maintenance support performance increases time used for maintenance operations and this decreases the level of availability.

## **2.2 Diagnostics and Prognostics**

Diagnostics is generally used with condition monitoring. IEC has defined the term fault diagnosis as ‘action to identify and characterize the fault’ [5, IEC ref. 192-06-20]. This definition has some differences when it is compared to out-of-dated versions of this term made by IEC. Alanen et al. have defined term diagnostics as ‘actions taken for fault recognition and fault localisation’ [2, p. 13]. Despite small differences between definitions, meaning of the term is almost same; to detect and understand the fault.

While diagnostics is for taking actions for fault recognition and fault localisation, prognostics is for estimating remaining useful life of the faulty unit. Remaining useful life can be calculated, and this helps scheduling maintenance operations.

With prognostics, it is also possible to calculate different outcomes if some changes are made to the unit. [8, p. 25-9] [16, p. 10, 12] For an engine, oil change could be this kind of change; with fresh oil the damaged engine could run longer compared to running the engine with the old oil.

## **2.3 Purposes of Condition Monitoring and Diagnostics**

As the definition of the condition monitoring says, condition monitoring is used to acquire information from the monitored system. Modern systems use automatic condition monitoring, where data is sent from the system to the computers for processing and analysing purposes [18, p. 7-8].

The diagnostics tool uses the data that condition monitoring has acquired. It processes and analyses the data, and based on the given rules, it indicates the status of the system and suggests actions for the detected faults by using prognostics.

With condition monitoring and diagnostics, the status of the system is known. It is possible to detect failures beforehand and schedule maintenance operations in advance. This is the reason why condition monitoring and diagnostics are used almost in every modern system; it can save money and time when used correctly. [8, p. 25-8] [18, p. 1] For experimental facilities like ITER, these tools are important, because otherwise acquiring data would be very difficult task, and the status of the system would be unknown. Scheduling the maintenance operations would be impossible without information of the system status.

## **2.4 Condition Monitoring and Diagnostics Methods**

Condition monitoring and diagnostics can be performed with two different ways. The first one is more traditional, artificial intelligence (AI) based method and the second one is more modern, model based method. [11, p. 2-3] [17, p. 390, 393] There is a significant difference between these two methods; following sections will explain main differences more detailed.

### **2.4.1 Artificial Intelligence Based Diagnostics Method**

AI based method, which is older one of these two methods, but still widely used method. This method is based on input-output measurements of the observed unit.

With these measurements, a data-based model can be created. Depending on the input values, corresponding output values can be found from the data. These estimated output values are compared to the real output values of the unit. [11, p. 2-3] [17, p. 393-395] For example, when an engine is running, temperature of it can be measured. Depending on the input values of the engine, there is an estimated value in the data for the temperature. This value is compared with the real value, and differences between these values can reveal faults.

Simplicity of the method is an advantage compared to more complex model based method. Creating rules for limit checking is quite straightforward operation. Rules are usually created with simple if-else logic. Disadvantage of the method is that the input-output measurements of the unit may often include errors and noise, and the dataset may be incomplete. [11, p. 2-3] [17, p. 393-395] These disadvantages can decrease accuracy of the method.

### 2.4.2 Model Based Diagnostics Method

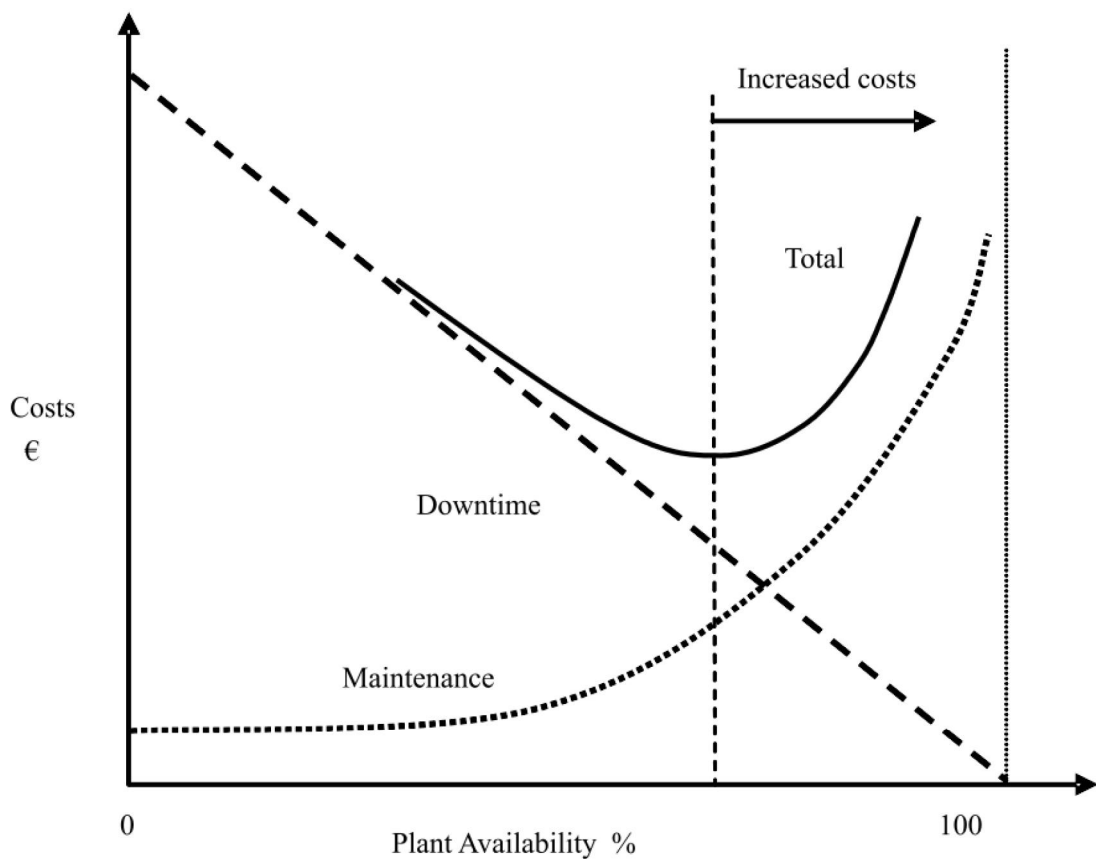
Model based diagnostics is more modern method compared to AI based method. Some kind of model of the monitored system has to be created in order to use the model based method. The model can be a logic based model or a model created with equations. The model is used to detect differences between the real system and the model, like in the AI based method. If the data coming from the system does not match with the verified computer model, there is a problem in the system.

The main difference between the AI based method and the model based method is how the output values are calculated or estimated. In the AI based method, the data is measured in advance. Estimated output values are based on the input-output measurements. With model based method, these values are calculated with the created model.

The model based method has one major disadvantage. In order to create accurate model, it may require a lot of equations and parameters. A model with high amount of parameters is difficult to create and tune, and data for some of the parameters may be unavailable. This disadvantage is pronounced when creating models of complex systems. [11, p. 2-4] [17, p. 390-391] If the model is inaccurate, it may generate false alarms. Due to this, the model may become unreliable and useless. On the other hand, successfully created model can be very accurate and provide very useful data for operators of the system.

## 2.5 Maintenance Strategies

Maintenance operations costs money. In Figure 2.2, it is possible to see how availability and total costs are affected by maintenance and downtime costs. If there is no maintenance and therefore no maintenance costs, total costs are high due to downtime costs. On the other hand, if there is too much maintenance, the total costs are still very high but availability is great. For example, in aerospace applications, risks must be as low as possible [18, p. 6]. This means high availability and high total costs. If the target is to achieve low total maintenance costs with decent plant availability, there is a compromise where downtimes are increased a bit and availability is decreased compared to aerospace industry levels, but total costs are minimised.



**Figure 2.2** An example how downtime and maintenance costs, and plant availability affect each other. Total costs are high if the availability is at maximum level. On the other hand, lack of the maintenance operations will also increase the total costs, because costs caused by downtime are high. [18, p. 5]

With different maintenance strategies it is possible to find suitable strategy for cov-

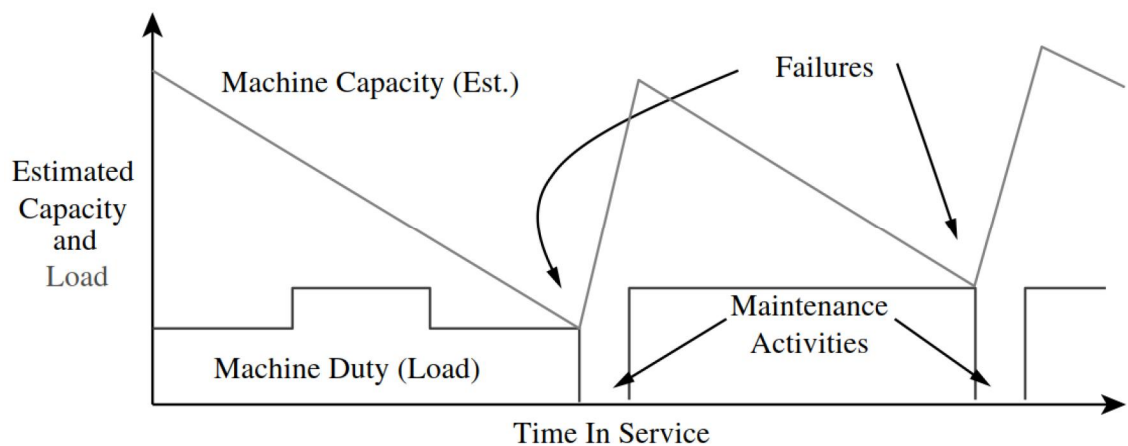


ering various requirements. Because of different types of machines and requirements, there exists different maintenance strategies for different purposes. There are three commonly used maintenance strategies in everyday use. These are called as a breakdown maintenance, a preventative maintenance and a condition-based maintenance strategy. [18, p. 5-9] Each one of them has advantages and disadvantages compared to each other. In the following subsections these maintenance strategies are introduced more accurately and how they are suitable for use in different use cases.

### 2.5.1 Breakdown Maintenance Strategy

Breakdown maintenance strategy is the simplest strategy of the three alternatives. It is also known as run-to-failure maintenance strategy. It is typical for this strategy that nothing is done until the unit fails. Maintenance costs are low, because there is no preventative maintenance before the breakdown. Most of the costs are caused by the maintenance activities and downtime costs after the breakdown. [8, p. 25-4 – 25-5] [18, p. 5-6]

The principle of this maintenance strategy is presented in Figure 2.3. In the figure, when capacity of the machine is decreased to the same level as machine duty, breakdown occurs. This is followed by maintenance activities, which takes some time. When the maintenance operations are done, machine is available again for operation until next breakdown. The machine duty may vary, which is presented in the figure with differently sized machine duty bars.

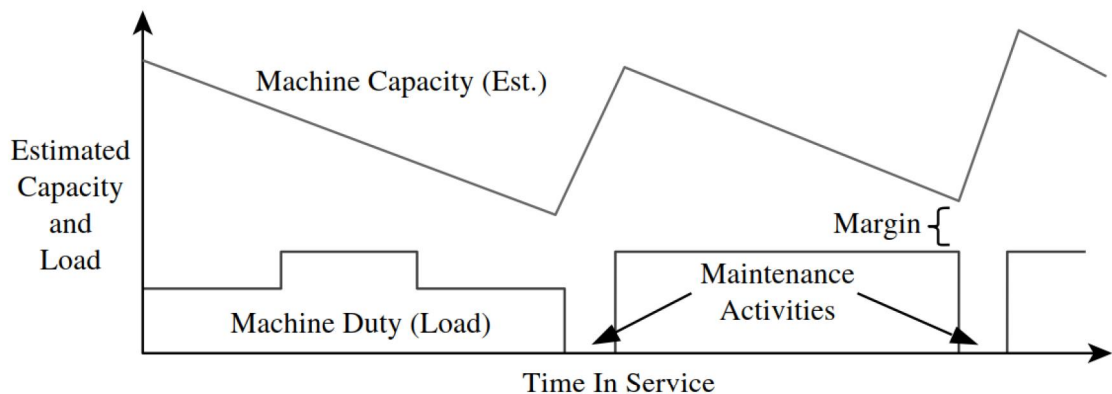


**Figure 2.3** A principle of breakdown maintenance. Machine capacity is decreasing during the time in service. When the machine capacity is equal with machine duty, failure occurs. After the breakdown, maintenance activities are taken to restore the machine capacity back to the original level. [8, p. 25-5]

The breakdown will be a surprise almost every time. The broken unit may be easy to repair or replace, but down times may be unacceptable. Breakdown maintenance is suitable for simple systems, in which down times do not cause safety risks or heavy economical losses. [18, p. 5-6] This strategy is suitable to be used for example with monitors of the control room at ITER. Broken monitor is easy to replace and it will not cause safety risks or increased costs. If there is no replacement monitor available, it may take some time to find one to replace the broken one. In the reactor building this strategy is almost useless due to its very high availability risk.

## 2.5.2 Preventative Maintenance Strategy

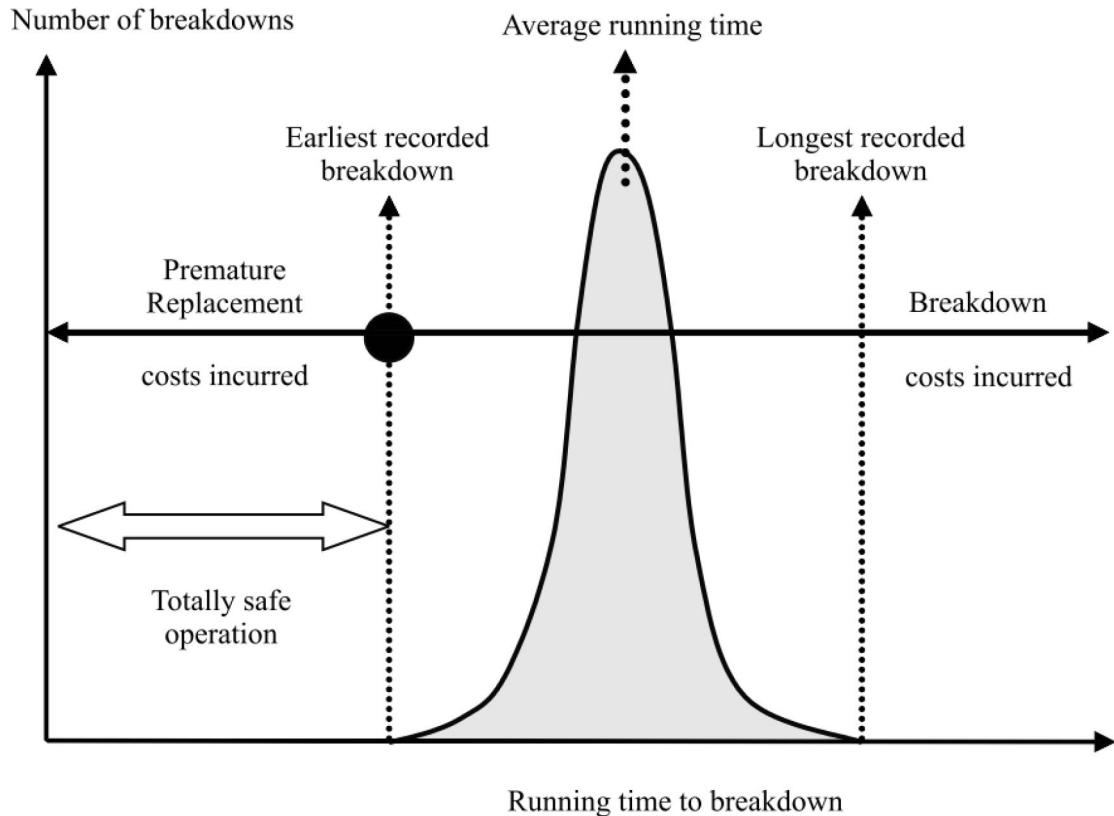
Preventative maintenance strategy is also known as schedule-based maintenance strategy [2, p. 44] and planned preventative maintenance strategy [18, p. 7]. Preventative maintenance strategy is an improved strategy compared to breakdown strategy. In the preventative maintenance strategy, replacing components is scheduled in advantage before there is a risk of failure. [8, p. 25-5] [18, p. 6-7] This principle is possible to see in Figure 2.4. Note the difference compared to breakdown maintenance strategy, in which there is no margin between machine capacity and machine duty before maintenance activities.



**Figure 2.4** Preventative maintenance strategy. Maintenance activities are scheduled so that there will not be breakdowns. When the maintenance activities begin, there is still machine capacity remaining. [8, p. 25-6]

Intervals between the unit replacements are based on manufacturers' data and user experience. This strategy is very common, for example, in aerospace industries. There failures are not allowed, so components will be replaced before the risk of the failure starts to increase.

The total maintenance costs will be high, because units are replaced often. This strategy needs reliable data of lifetime of the units. If there is no data available, it is difficult to know when each unit should be replaced. If the unit is replaced too often, it will increase total costs, and if it is replaced too late, it may cause increased safety risk. [8, p. 25-5] [18, p. 6-7] This is illustrated in Figure 2.5.



**Figure 2.5** An example of unit failure distribution. Preventative maintenance strategy requires this kind of information for scheduling of maintenance. [18, p. 7]

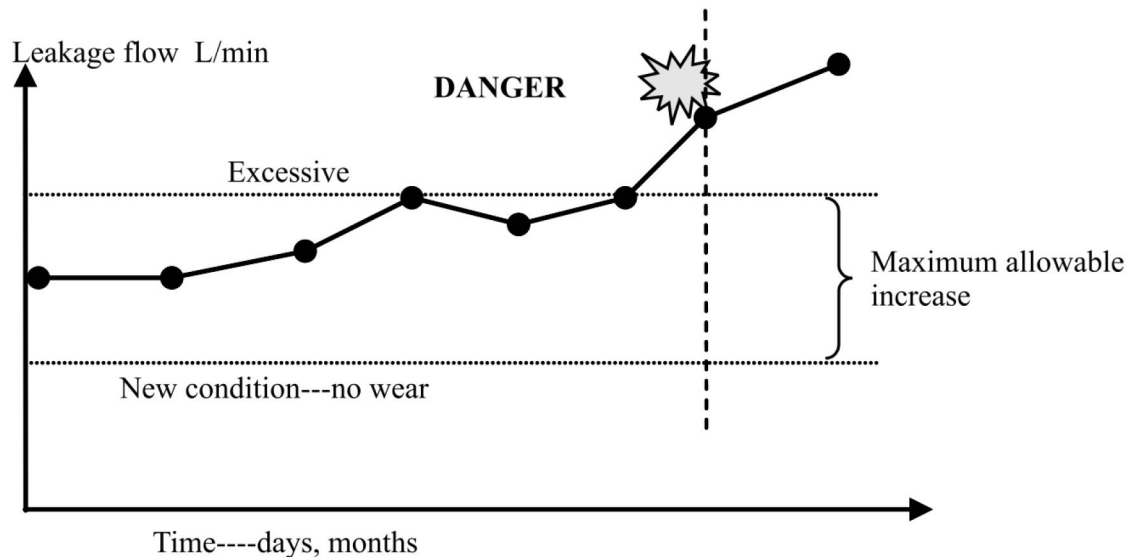
At first, there is totally safe operation time interval in the beginning. After this point, breakdown may occur and the unit reaches the average running time. After this point longest recorded breakdown time is coming close and the risk for unit breakdown increases all the time. In aerospace applications the components are replaced at earliest recorded breakdown point in order to minimize possible risks. Increased maintenance costs must be accepted when using this maintenance strategy.

### 2.5.3 Condition-Based Maintenance Strategy

Third maintenance strategy is condition-based maintenance strategy (CBM). This strategy is more focused on health of the system instead of predetermined schedules

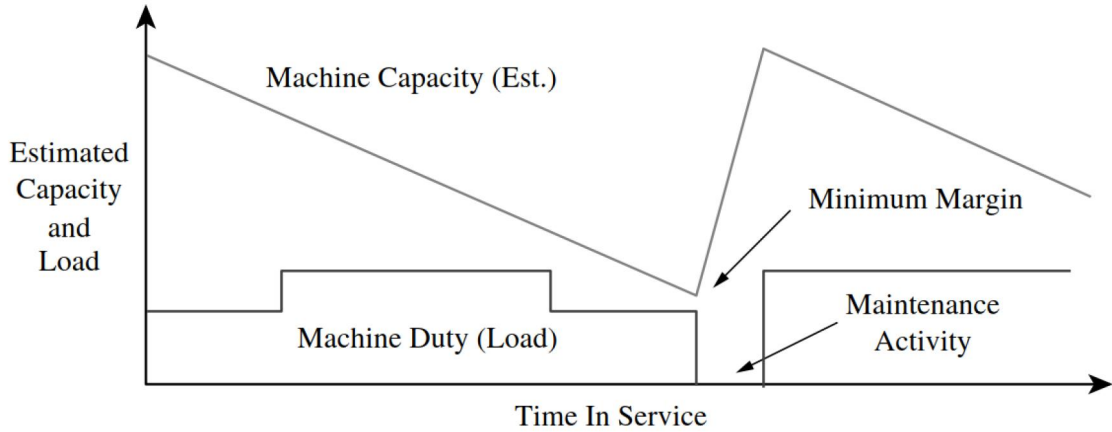
like in the preventative maintenance strategy. The margin between the machine capacity and the machine duty is much smaller compared to preventative maintenance strategy. The difference is possible to see in Figure 2.7. In order to monitor health of the machine, condition monitoring operations are required for analysing the state of the unit. [18, p. 8][2, p. 40]

With CBM, the status of the machine is known all the time. Unlike in preventative maintenance strategy, maintenance operations are performed only when required. If there is no signs of wearing or faults, there may be no reason for replacing the unit. When there are first signs of problems, maintenance operations can be scheduled in advance. The main principle of this maintenance strategy is presented in Figures 2.6 and 2.7.



**Figure 2.6** An example of condition-based maintenance strategy. Condition of the unit is monitored all the time. When the monitored value is out of the limits, maintenance operations for the unit can be scheduled and the unit will be replaced at suitable moment. The unit should be changed before reaching dangerous values. [18, p. 8]

As seen in Figure 2.6, there are maximum limits for values. If the unit exceeds one of these limits, the operator will get a warning. The operator can now decide what to do and schedule maintenance operations for the unit. The unit can be in use after the warning but actions should be taken soon.



**Figure 2.7** When signs of problems starts to occur, maintenance activities are scheduled before breakdown happens. There is no breakdown, because minimum margin is still remaining. [8, p. 25-7]

With this maintenance strategy, it is possible to avoid unnecessary maintenance operations. This will decrease maintenance costs. On the other hand, creating a system for CBM strategy may become expensive. Big facilities may require a lot of sensors, other hardware and software. It is also more complicated compared to two previously mentioned maintenance strategies. [8, p. 25-6 – 25-7] These facts will increase initial maintenance costs, but once the CBM system is created, it will no cause as much costs as in the beginning.

This strategy is suitable for facilities like ITER, in which no data is available of lifetime for different components. This strategy will provide support for scheduling the maintenance operations by providing information about the current state of the system. [13]

## 2.6 Suitable Maintenance Strategies for Remote Diagnostics Application

The Remote Diagnostics Application will be based on preventative maintenance and condition-based maintenance strategies. With the application, the focus is on preventative maintenance strategy strategy, because it provides most accurate data and status of the system and single units. At ITER, the main strategy is preventative maintenance strategy, and the CBM strategy will support scheduling the maintenance operations [13].

With the application, the preventative maintenance strategy will provide support for failure detection rules. For some components, it is known how long they can be used before breakdown occurs. This data can be used together with CBM to make correct decisions for failure detecting. Example of this kind of device is hydraulic filter, which will clog during the time. With CBM, it is possible to measure the real condition of the filter. By combining this data with preventative maintenance strategy, accurate enough approximation of the remaining lifetime can be calculated and maintenance operations can be scheduled.

AI based and model based diagnostics are both used with RDA. With AI based diagnostics, thresholds and limits can be monitored easily, if measurement data for the unit is available. Model based diagnostics can be used for more complicated diagnostics purposes, for example detecting water leaks from the system. By creating a model, it is possible to calculate, how much there should be water in the system, and by comparing the calculated values to the real system, leakages can be found [13].



## 3. DEVELOPMENT OF THE REMOTE DIAGNOSTICS APPLICATION

Remote diagnostics is a part of the remote handling systems of ITER. The maintenance equipment is fully remotely controlled and therefore also remote diagnostics is used. In this chapter, it is described, why creating the Remote Diagnostics Application is required and what it should be capable to do. In addition, the development process of the RDA is described in more detailed. The focus is on the development process of the RDA front end implementation, which was the main task of the thesis.

There is also a four part standard ISO 13374 ‘Condition monitoring and diagnostics of machine systems. Data processing, communication and presentation.’, which provides the basic requirements for open diagnostics software specifications. This standard is useful to use with this project and it is introduced in this chapter. The standard will be also referred later in this thesis in Section 5.2, where implementation of the prototype version of the RDA is compared to this standard.

### 3.1 Remote Handling Systems

Because of radiation and unpleasant conditions in the ITER reactor, all maintenance operations are performed remotely controlled. There will be a separate control room for these operations. An example of this kind of control room is presented in Figure 3.1. The control room of the figure is located at VTT for controlling Divertor Test Platform 2 (DTP2), which is a real size mock-up section of a vacuum vessel bottom of ITER. This kind of set-up allows a safe working environment for the operators. Technology is similar compared to space located missions, where controllers are located on earth and the spacecraft is flying far away in the space. Distance does not matter, only data transfer must be possible between control room and the equipment. [9] [6]

Virtual reality (VR) platforms, powerful computers and advanced technology should make the remote handling operations of ITER easier and more effective. With simulations and the VR systems, the operator can virtually see the maintenance

operations and the equipment despite physical absence. [9] This kind of operation requires very accurate and real-time simulations. In Figure 3.1, it is possible to see how the DTP2 is controlled from a control room with the real-time simulations without possibility for direct visual contact to the system.



*Figure 3.1 DTP2 control room at VTT. The remote handling operations are possible with the accurate simulations and virtual reality systems based on data acquired from the real equipment. [6]*

With the remote control and remote diagnostics systems, the main idea is the same: operators are working remotely without the visual contact to the equipment they are working with. Because the system is remotely controlled, it is important to get a feedback from the system in order to know its status. There might be a fault in the system, but the operator can not detect it without the condition monitoring. Visual inspections are not possible with this kind of remotely controlled systems.

### **3.2 The Reason for Developing a New Remote Diagnostics Application**

The diagnostics applications are not a new thing in industry and they are used for several purposes. Because they are not a new thing, why is it required to create a new application instead of using complete solutions with the remote handling systems of ITER?



There are many different diagnostics applications for different purposes available on the market. Many of them are designed for specific purposes. As examples of these are applications designed for monitoring statuses of bearings and motors. This kind of diagnostic applications offer very accurate and highly automated diagnostic services for mass-produced devices and components. [12] The bearings are very common and simple components in different machines and one specified application can monitor them easily.

The diagnostics application for ITER represents a very different approach. The components and devices are individuals, and previously mentioned specific applications are not compatible with this kind of system with many different components. The components and devices of the system may be improved and changed during the development process of ITER, so the application should be customisable enough to meet the variable and flexible requirements. [12] In order to meet the customer requirements, a new remote diagnostics application is required to be developed for the ITER remote handling systems. By creating a new application, it is possible to create an application with features that are required without compromises. Disadvantage of creating a new application is a long development time and costs caused by development process.

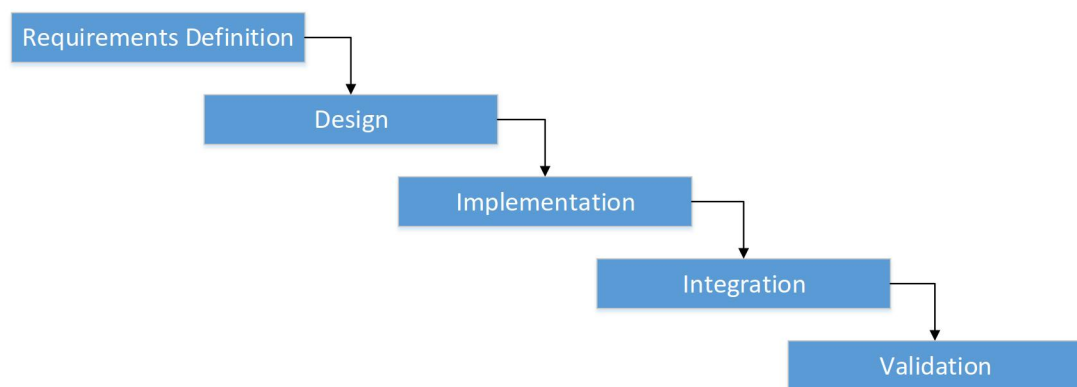
### **3.3 Development Process of the Remote Diagnostics Application**

When the writing process of this thesis started, there was already some components of the application under development and plans were made for future development process. The primary task of this thesis was to develop the front end implementation. This included architectural solutions for graphical user interface (GUI), GUI itself, different monitoring components and integration of different components of the application to the GUI.

Software development process may be very complex. This depends partly on the used life cycle model. With the models, the project should be more easy to control and understanding the current state of the project should be easier. For software development processes, there is a great variety of different development models available. The best known models are waterfall and spiral models. [4] These models are described more detailed in the next sections.

### 3.3.1 Waterfall Model

In the waterfall model, the development process has clear stages. Typical stages are requirements definition, system design, implementation, integration and validation processes, which also are used with this project [1]. When one stage is finished, the process continues to the next stage. There might be small modifications made to the model, depending on the type of the project. One example of the modifications is, that there might be slight overlapping between two stages, which allows revisiting in the previous stage. [4] The principle of the waterfall model is presented in Figure 3.2.



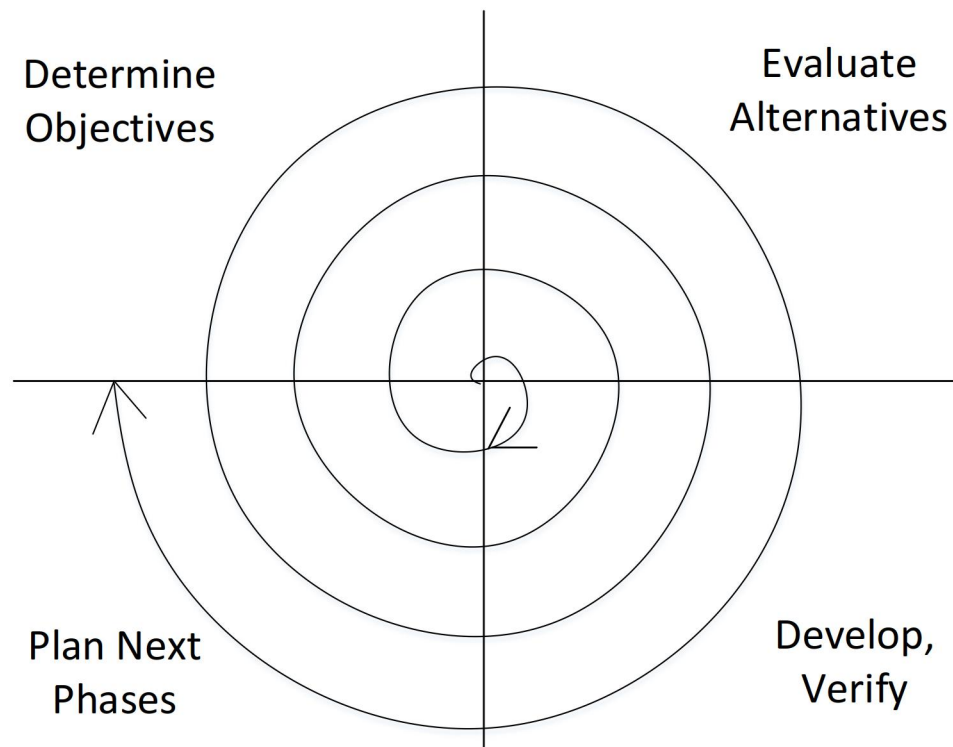
*Figure 3.2* A principle of the waterfall model. The development process starts from top of the waterfall with a requirements definition. After completing a stage, the process continues to a lower stage, until it reaches the last stage of the waterfall. [4]

The waterfall model is easy to understand and to follow. It has one major disadvantage, which is inflexibility. The model does not normally allow revisiting the previous stages. If there is a problem, which is caused by poor choice in earlier stage, it can not be fixed easily. The waterfall model is usually used with big projects. [4] Each stage is planned beforehand and documented properly, which makes budgeting easier [1].

### 3.3.2 Spiral Model

The second well-known development model is the spiral model. In the spiral model, the development process is like in the waterfall model, but there are several iteration rounds. Reference model for the spiral model is presented in Figure 3.3.

With the spiral model, the development process starts with determining objectives. This is followed by evaluating alternatives, development process and finally planning next phases. When the first round is completed, the process is started again and again, until the task is completed. After every round the process is moving towards the end. Because every round consumes time and resources, the costs of the project will increase at every round. [4]



**Figure 3.3** The spiral model. In the spiral model, the development process includes several iteration rounds. After each round, the project should be more completed. [4]

The spiral model is more flexible compared to the waterfall model. Major problems can be found earlier during the development process compared to the waterfall model. [4] With this model, the project can be completed in smaller tasks and improved versions can be created after every round. This allows more flexible software development process. The disadvantage is, that it is difficult to forecast how many iteration rounds is required and due to this, budgeting is also difficult.

### 3.3.3 Models Used with Development Process of the Remote Diagnostics Application

The project for creating the RDA is using the waterfall model [1]. Before the writing process of this thesis started, the requirements definition and design phases were completed. When writing of the thesis started, the project moved to the implementation phase.

The waterfall model is used for the project, because it provides clear phases for different tasks of the project. With this model, also the budgeting of the project is easier, and this was one of the main reasons for choosing this model. [1] There are documented plans for the project and it is more clear for the customer, in which phase the project is currently running.

The implementation process is one large, single stage in the waterfall model. This stage can be completed by combining different models. The main model for the process is the waterfall model, but the stages of the model can include different models for completing that one phase. For example, the implementation stage can be considered as a single spiral model. During that phase, different implementation options are evaluated. A piece of code is created, and after testing, it may require changes. Making changes is easy operation with the spiral model, because it allows several iteration rounds for the code. The requirements of the RDA were not very strict and allowed different alternatives to be tested, and for this kind of flexibility, the spiral model offers better iteration abilities compared to the waterfall model.

## 3.4 Development Platform

The application will be developed with the graphical LabVIEW development platform. There were different alternatives for LabVIEW, but in a comparison made by VTT, the LabVIEW was chosen as a best alternative. More about this comparison and alternatives is described in Subsection 3.4.2.

LabVIEW is an abbreviation of the words Laboratory Virtual Instrument Engineering Workbench. The software is developed by National Instruments, which is relatively big company in USA. The software is marketed as systems engineering software, which simplifies hardware integration allowing easier data acquisition operations. One of the purposes of the software is that it reduces complexity of programming, making it easier and faster. [19] [20]

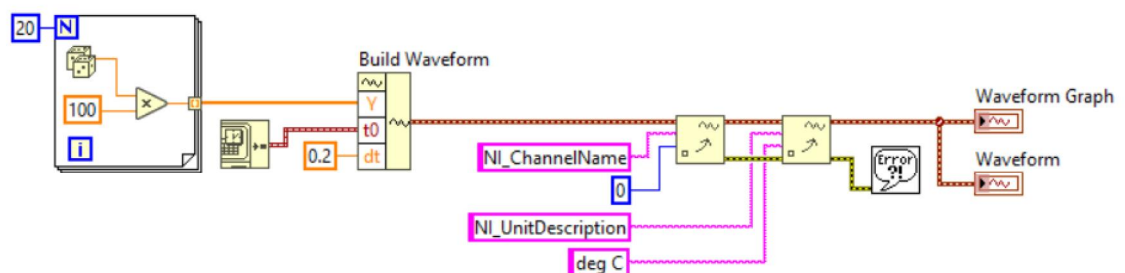
First version of the LabVIEW software was released in 1986 as a tool for scientists

and engineers for performing automatic measurements [19]. The latest version of the software is version 16.0 (Fall 2016) and the prototype of the RDA will be developed with this version. New versions of the software is coming out during this project and it is possible that upcoming versions of the RDA will be developed with one of these new versions.

### 3.4.1 Development with LabVIEW

LabVIEW is a graphical programming software. The main idea with the software is to use different blocks and wire them together, which is also called as dataflow programming or dataflow language. [19] Each one of the provided blocks has a different function. For example, a block for multiplying operation has at least two inputs and one output. The block performs multiplying operation for all the inputs and the result is sent to output. The code for the application is done by combining different blocks together with wires. The software includes many different blocks for use and more can be downloaded as an add-ons.

Figure 3.4 illustrates an example of creating a waveform graph with attributes with LabVIEW. At first, there is a for-loop, which generates 20 random numbers multiplied by 100. After this the waveform is build with created data points (Y), current time (t0) and time difference between each sample (dt). Then two different attributes are added in order to include some extra information to the waveform. Finally there are blocks for visualising the created graph and attributes. Also one error block is added for an error handling operations.

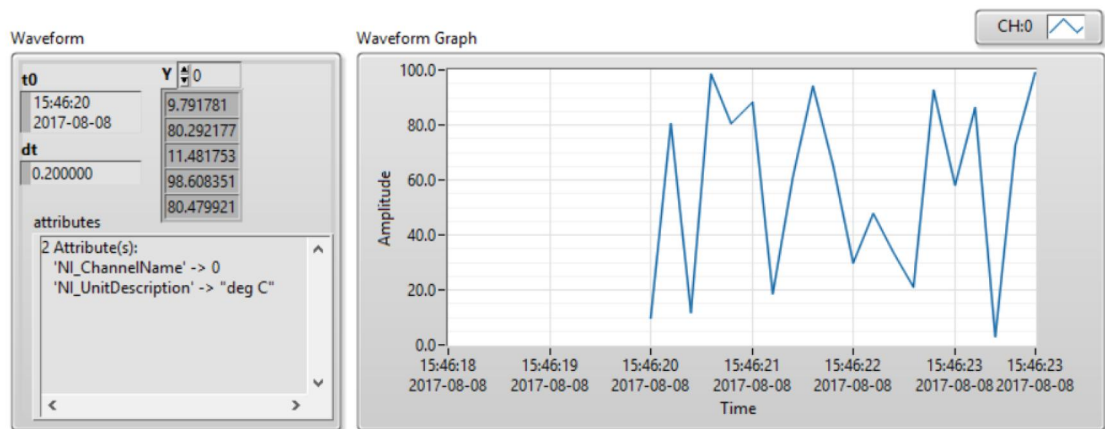


**Figure 3.4** An example of LabVIEW code. The code generates a waveform chart with optional attributes (*NI\_ChannelName* and *NI\_UnitDescription*). This example is available in LabVIEW build-in examples (*Waveform - Create.vi*).

Figure 3.5 presents the front panel of the previous example. The waveform graph includes the data points and the time when the data points are created. On the left,



the extra information is visible, including all the data points, the first time stamp, the time difference between each sample and the attributes.



**Figure 3.5** The front panel view of the example code. Generated waveform graph is visible on the right and the attributes and values on the left.

LabVIEW uses the .vi file format, which is abbreviation for Virtual Instrument (VI). A .vi file consists of two parts: front panel and block diagram. [19] The front panel is the user interface for the code. The block diagram includes all blocks and code made for the file. These two parts are always connected to each other. Every control and indicator has its counterpart in the front panel and in block diagram. User interfaces can be done easily to the front panel. Examples of the front panel and the block diagram are in Figures 3.5 and 3.4 respectively.

LabVIEW is marketed as a user-friendly development software, which is achieved in many ways. By using block diagrams, the created code is easy to understand and follow. In addition, LabVIEW is compiling the created code in real time. This allows finding errors easily, because the software will instantly report syntax errors found in the code. The software indicates the block where the error occurs and what the problem is.

Every available block has a help file, which makes it easy to understand how to use different blocks. The software has also many different examples to demonstrate, how to use different structures and blocks. Some structures are provided as templates, which are helpful, if the user does not have previous experience of the structure.

### 3.4.2 Alternative Development Platforms

There are alternative development platforms for LabVIEW. VTT has made a comparison, in which different development platforms from different developers were compared before the developing process of the RDA started. The selection criteria for the development platform were known before the comparison was made. IoT - Ticket from Wapice, Wedge from Savcor, MUST from European Space Agency and LabVIEW from National Instruments were the participants in this comparison. These platforms were studied as to how they are able to satisfy the selection criteria, and based on the suitability, the platforms got a grade for every requirement. Each criterion had a different weight factor. With grades and weight factors, the ranking list for the platforms was created.

None of the softwares did meet all the requirements, which was expected. LabVIEW was reviewed as the best option for the development platform. It was on top of the ranking list and it has been in use at VTT in previous ITER and Fusion for Energy projects, so it was known how it can be used and what are the capabilities of it.

Developing a very new development platform was not an alternative. A new software takes a decade to mature and the project does not have time for that. A problem with the new software is always the amount of bugs and possibly missing features and usability. Benefits of the new development platform would have been that it would cover all the requirements, because it is created only for this project. Disadvantages are still greater compared to the benefits, so no new development platform was created. [10]

## 3.5 Requirements for the Remote Diagnostics Application

The requirements for the application were set by the customer. The project has a great variety of different requirements and those are the baseline for the development process of the application. There are requirements for every component and feature. This thesis is more concentrated on the front end implementation, so the requirements for that part of the application are introduced in this section. The requirements are listed in an internal, unpublished document.

The application is rule-based, and due to this, most of the front end implementation related requirements are based on requirements of the rule system. There are many different requirements for the rules. For example, the user should be able to create and edit these rules and select, which ones are active. The rules are used to determine

when there are prevailing or upcoming faults in the system. Based on these rules, warnings and alarms are created to inform the user about the faults. An example of a rule is a limit checking of the signal; if the signal reaches limits, an error or an alarm is created. These alarms and warnings required indicators to the front panel in order to provide this information for the user.

Due to the requirements for creating and editing the rules, one major requirement for the application is that it should be dynamic. Because the application will be generic and compatible with novel diagnostic cases, there can not be many fixed features and rules for signal processing. In addition, the application should always be running when the maintenance equipment is on line, so it is not possible to recompile the application if new rules or primitives are added to the application. The application should also be very reliable in order to achieve a good usability.

Because the requirements do not include strict definitions as to how the requirements should be covered in the application, there are possibilities for different implementation options. The architectural choices for the front end implementation was one of the most important thing in order to cover requirements of dynamic behaviour. A more detailed description about the application architecture is presented in Chapter 4. [14]

### **3.6 ISO 13374 Based Guidelines for Development Process**

ISO 13374 standard ‘Condition monitoring and diagnostics of machine systems. Data processing, communication and presentation.’ provides basic requirements and guidelines for condition monitoring and diagnostics softwares. The standard includes four parts; ‘Part 1: General Guidelines’, ‘Part 2: Data Processing’, ‘Part 3: Communication’ and ‘Part 4: Presentation’. This subsection is mainly based on the first part, because it provides most of the useful general information for the project. Other three parts have more specific information about their topics and they are not discussed in this thesis.

At first, the condition assessment blocks are introduced. These blocks are for describing the functionalities of the application. Secondly, there are descriptions of monitoring features, which are made for the operator for receiving information and messages from the application.



### 3.6.1 Condition Assessment Blocks

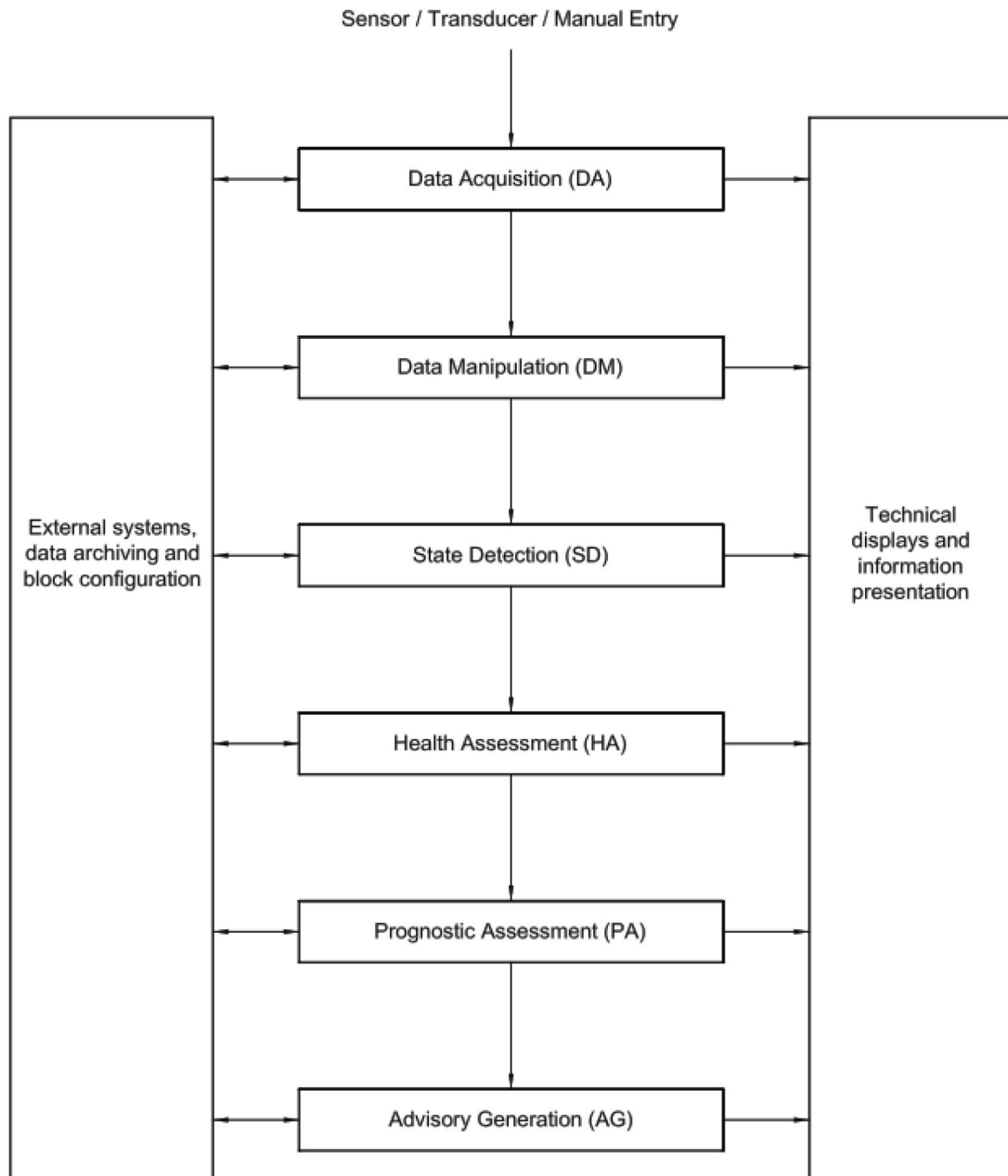
The ISO 13374 part 1 provides a useful reference model for diagnostics information progress in the reference application. This model is presented in Figure 3.6. In the figure, the information progress starts from the top of the model.

The first block is called as data acquisition, which converts the data to a digital parameter. This parameter represents a physical quantity and information related to it, which can be, for example, time or data quality. The second block is for data manipulation, which performs simple signal analysis. It also creates virtual sensor readings based on raw measurements. State detection is the third block, and it is responsible for detecting if the data is in warning or alarm area.

The fourth block is called as health assessment, which diagnoses the health of the system. Prognostic assessment is the fifth block and it is responsible for trying to forecast the remaining useful life of the units. These forecasts are based on history and current live data. The last block is for advisory generation, which gives additional information of faults and status of the system.

The three first blocks are technology-specific blocks. These blocks requires functions, which are targeted to a particular technology. Examples of these technologies are shaft displacement monitoring and bearing vibration monitoring. With the RDA, no specific technology is used, because the architecture of the application is more flexible compared to this reference model. The last three blocks are for monitoring the health of the system, predicting remaining useful life and providing tips for recommended actions. [16, p. 1-3]

The standard also includes blocks for the external systems and technical displays. Each of the layers, except the advisory generation block, will be implemented in the prototype version of the RDA. This project, especially the front end implementation, focuses on state detection and health assessment stages. The rules of the application will cover both of these blocks. The prognostic assessment is also important part of the application, and the role of it will increase in the upcoming versions of the application. More about implementations and features of the application, compared to this standard, is described in Chapter 5.2.

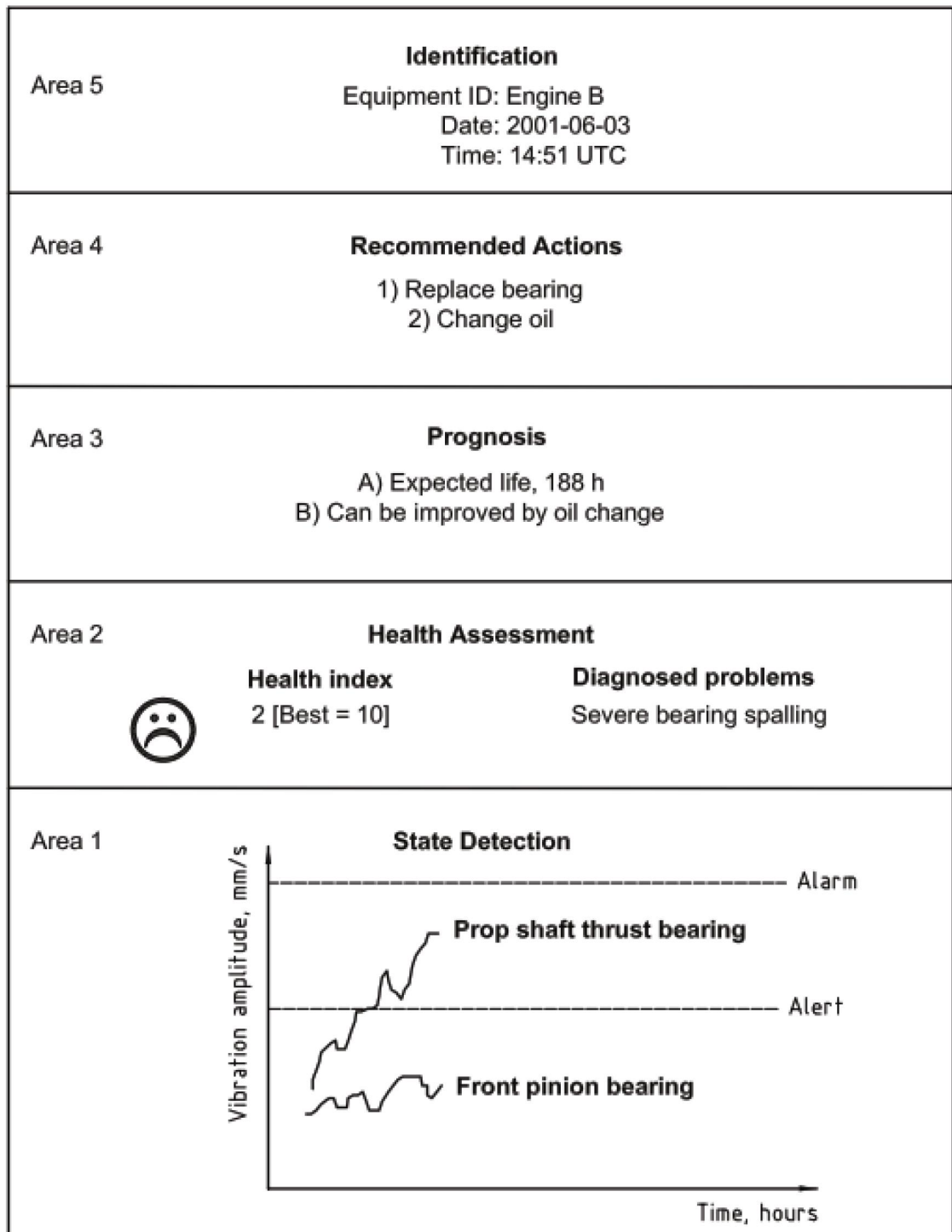


*Figure 3.6* Diagnostics software architecture based on ISO 13374 standard. Sensor data is coming from top of the diagram and it is processed in each block and external systems. The technical displays are for presenting the processed information. [16, p. 2]

### 3.6.2 Monitoring Features

Architecture of the software is important for the developers of the RDA. The operator is more interested about usability of the application. In order to provide information for the operator, indicators are required. For this purpose, the standard provides a good reference model of what kind of information and how it is

recommended to show for the operator. This model is presented in Figure 3.7.



*Figure 3.7* An example of monitoring features based on ISO 13374 standard. The information is processed in each block and displayed for the operator. Each area has different kind of information to show for the operator. [16, p. 12]

As illustrated in the previous figure, the front end of the application is divided into distinct areas. Area 1 includes signals and thresholds visualised in a graph. This area is also called as a state detection. With this simple graph, the operator can see the values of the signals and how close they are to the limits.

Area 2, which is also known as health assessment, includes information about the health of the system. This area indicates for the operator the status and condition of the system. The health assessment uses rules for monitoring the health. In this example, the state detection indicates the prop shaft thrust bearing is closing to the alarm level, so the health assessment indicates the health index is very bad.

Area 3 includes prognostics for the system. Based on the data of the bearing, prognostics calculates expected life for the bearing, and also gives advice what can be done in order to increase the expected life. With the prognostics, the maintenance operations can be scheduled and operation of the system continued.

Area 4 includes recommended actions in order to avoid the problem or to fix it. Area 5 includes more specific identification data of the faulty device or component and time and date when the problem occurred. [16, p. 9-10]

Each of these areas are going to be included to the RDA. In the prototype version, some of these areas may still be in very unfinished state, but the idea is planned to be included. In the next chapter, these features are described better as to how they are implemented in the application or how they should be working in the upcoming versions of the RDA.

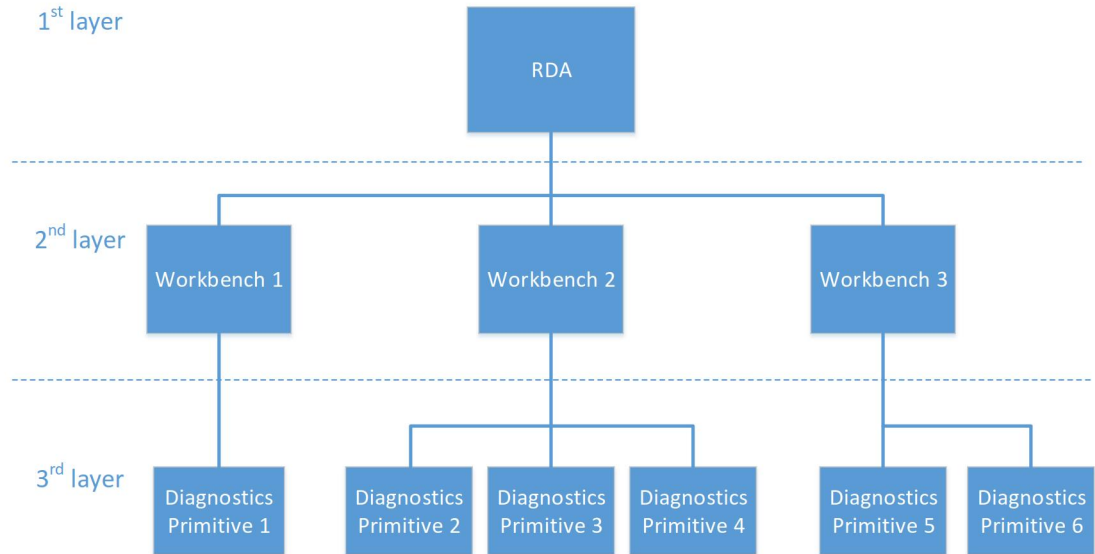
## 4. IMPLEMENTATION OF THE REMOTE DIAGNOSTICS APPLICATION

The main task in the thesis was the front end implementation and to create architecture for it. The front end includes the graphical user interface, which is the part the operator will see and use. The prototype version of the front end implementation of the RDA is presented in Appendix 1.

Front end implementation is created with three-layered architecture, which consists of workbenches and diagnostics primitives. In addition, there are different monitors for indicating important messages created by the application. The implementations of these components and the front end architecture are described in the next sections.

### 4.1 Three-Layered Architecture

The main architectural decision with the application is the three-layered architecture. The architecture was invented by VTT and implemented in this thesis. With this architecture, the requirements about dynamic and flexible structure of the application can be covered. Because the maintenance equipment of ITER is evolving all the time, and the equipment and the requirements may change, the RDA should be able to cover these changing situations. If there is some changes in the equipment, the changes should also be reflected to the application, without recompiling or aborting the application after making changes in the code. The three-layered architecture is visualised in Figure 4.1.



**Figure 4.1** The principle of the three-layered architecture. The main application is the first layer and the workbenches are in the second layer under the first layer. In the third layer are the diagnostics primitives. The amount of the workbenches and the primitive may vary.

The main idea with the three-layered architecture is, that there is three layers in the application. The top level of the application is constantly running, and there should be no need to abort it. This top-level is fixed and does not need changes if there is some in the monitored equipment. The top level can be considered as the main window, which provides framework for the application.

The second layer is called as a workbench layer, which includes the workbenches. The workbenches can be considered as sub windows in the main frame. Each workbench collects user defined diagnostics primitives into one sub window. There can be several of these workbenches under the top level layer, but at least one is required.

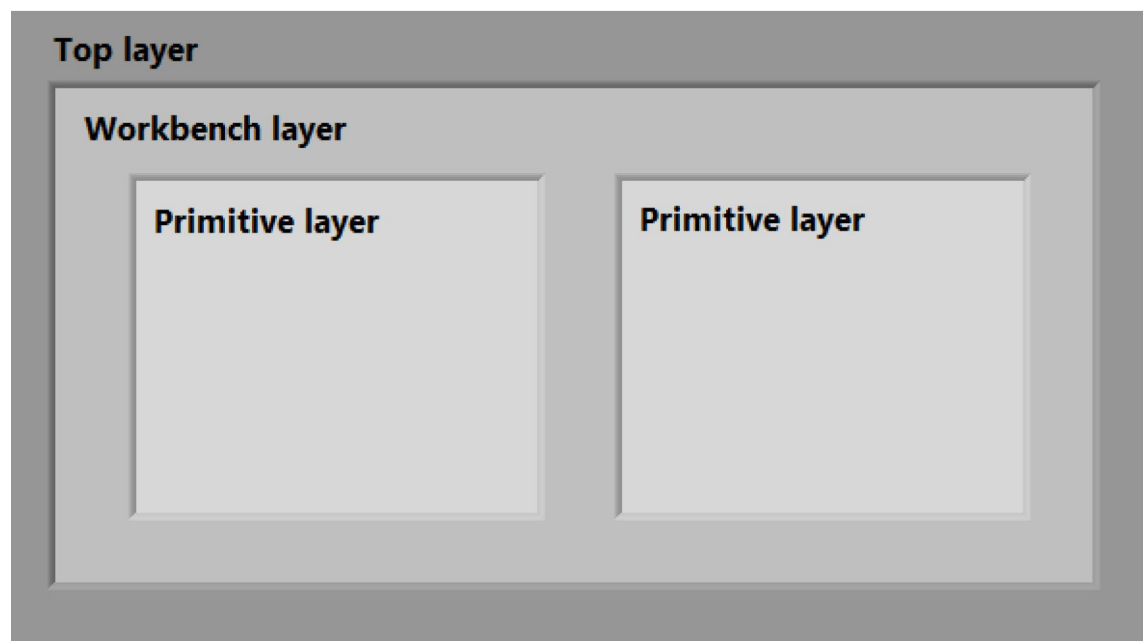
The third layer is called as diagnostics primitives layer. This layer is running under the second layer. Every workbench should include at least one diagnostics primitive. There is no maximum number for primitives but the user interface may limit this number, depending the space required by primitives on the screen. More detailed descriptions of the workbenches and the diagnostics primitives are included in Sections 4.2 and 4.3.

This architectural solution allows creating and editing workbenches and primitives even if the main application is still running. For editing or creating a new primitive,

only that one workbench, which includes the primitive, needs to be aborted. When aborting a workbench for editing operations, all the primitives under that workbench are aborted. If the top level is aborted, the whole application stops running. For example, if workbench number 2 of the figure is aborted, diagnostics primitives 2,3 and 4 are aborted as a result of this.

### 4.1.1 Sub Panels

In the LabVIEW development platform, the three-layered architecture is implemented with sub panels. The top level layer has sub panels for each workbenches, and every workbench has sub panels for each primitives. Simplified LabVIEW implementation of the three-layered architecture is presented in Figure 4.2.



**Figure 4.2** A simplified implementation of the three-layered architecture presented in LabVIEW development platform. The diagnostics primitives are included in the workbenches and workbench is included in the top level layer. This example has only one workbench and two primitives.

In the figure, the top level layer has one sub panel for the workbench. The workbench is a separate file, which is running inside the top level layer in the sub panel. The workbench has two separate primitives, and the primitives are also separate files running in the the sub panels. In this example, there is totally four separate files running: one top level layer file, one workbench file and two primitives. Every file



can be run independently, which allows flexibility for testing operations during the application development process.

In the prototype version of the application, the sub panels are not easy to detect, because the frames of the panels can be hidden and colors of the panels can be the same. This is intentional, because the end user of the application does not need this kind of information about the architecture. In the example figure, the frames and the colors are purposely noticeable in order to demonstrate the sub panel structure in the LabVIEW development environment.

### 4.1.2 **Dynamic Calls**

In the LabVIEW, the VIs can be loaded and called statically and dynamically. When using the static VIs, the VIs are loaded into memory at the moment when the main VI is started. These VIs are running as long as the main VI is running and those can not be aborted without aborting the main VI.

The second option to call and run the VIs is to call them dynamically. This feature is very useful with the sub panels and with the three-layered architecture. This allows running the VIs when requested from the main VI. With this feature, it is possible to open and close the workbench and the primitive VIs only when necessary. This allows aborting the VIs for editing purposes and run them again after editing without aborting the main VI.

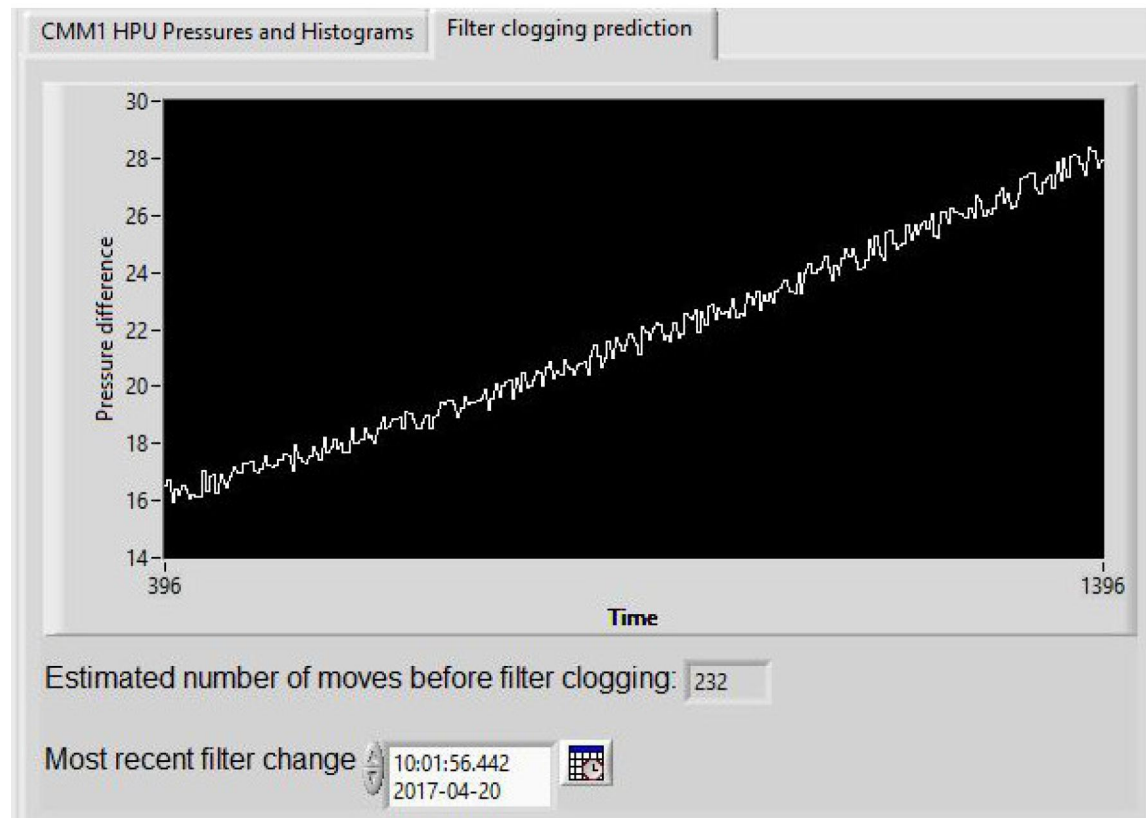
With the dynamically called VIs, the visible VI in the sub panel is possible to switch to another one. For example, if there are diagnostics primitives 1 and 2 running in the simplified application, which is illustrated in the Figure 4.2, it is possible to abort primitive 2 and to open and run primitive 3 in the same window without making changes to the code.

Closing unnecessary files also releases resources of the computer. Every running file consumes the processor power and the available memory. Impact of a single VI to the performance may not be noticeable, but if there are many unnecessary VIs running on the computer, the impact may be significant. The static VIs are always running on the computer and consuming resources, which is the biggest difference when the static VIs are compared to the dynamically running VIs.



## 4.2 Workbenches

As mentioned earlier, the workbenches are in the second layer under the top level layer. The main idea with workbenches is to collect different kind of diagnostics primitives into one window. Workbench itself does not have any monitoring features, because it is used only to collect primitives. The operator can decide what primitives are included in the workbenches. This behaviour is implemented with a RDA configuration file, which includes the workbench and primitive hierarchy. Example of a workbench including two primitives is presented in Figure 4.3.



**Figure 4.3** A workbench including two primitives in separate tabs; CMM1 HPU Pressures and Histograms, and Filter Clogging Prediction. Unit for the pressure difference is a bar and for time the unit is a second.

A workbench template file will be provided for the operator or third-party programmers in order to create new workbenches. Creating a new workbenches is simple, because the file can be reused without making any changes to the code. The workbench file uses the RDA configuration file, which includes all the information of what primitives each workbench uses.

## 4.3 Diagnostics Primitives

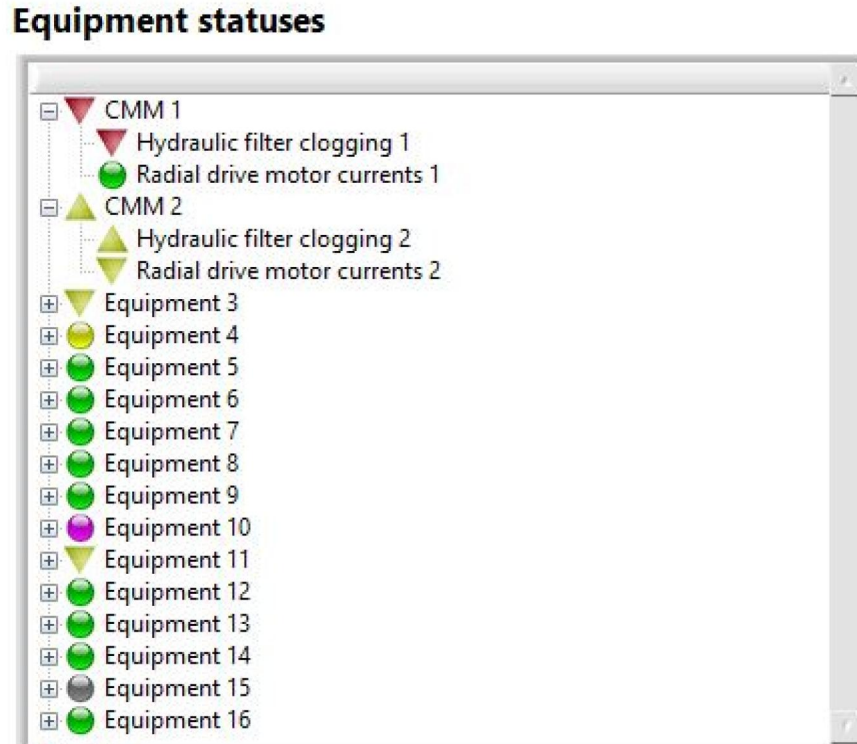
A primitive includes functionality to analyse and manipulate the data which the application is using for the diagnostics. A primitive can include almost anything; it can be used for monitoring a specific signal for filtering or any other processing functions, doing searches from archive data or creating plots based on the acquired data. One primitive can include many different features or many features can be distributed between several different primitives.

An example of a simple primitive is included in Figure 4.3 in the previous section. This primitive is monitoring a pressure difference of a hydraulic filter. It is also calculating an estimate of remaining useful life of the filter with a prognostic rule.

The plan is to provide a primitive template file for the operator. It is not as easy operation as providing the workbench template file, because there can be so many different types of primitives required. In order to access to the data that is required for the primitive, many different data interfaces needs to be provided. The data may be processed with different ways in each primitive, so creating one template file may be challenging.

## 4.4 Status Monitor

The monitoring features are one major requirement for the application. Because the monitoring of the system should be easy, a status monitor was implemented. Purpose of the monitor to collect the statuses of all equipment into one monitor, which is located at top level layer and is always visible. With this monitor, it should be fast and easy to see if there are warnings or alarms in the system. The prototype version of the status monitor is presented in Figure 4.4. The same monitor is visible in the picture of the prototype in the Appendix 1.



*Figure 4.4* The status monitor. The monitor is implemented for collecting the equipment statuses into one window.

The statuses are indicated with a symbol and a color. One equipment can include several different statuses. By opening the tree, the operator can see all the statuses of different devices and see where the problem is. In the example, CMM 1 includes two items to be monitored. The top level status is indicating the worst case status of all devices. In this example, the CMM 1 has red arrow down, because the ‘hydraulic filter clogging 1’ primitive has an active alarm. The other device of the CMM 1 has green symbol for indicating normal operation.

In the application, there are different symbols used to indicate statuses and severity of the events. With a symbol, it is easier and faster to detect the status of the equipment instead of text. Different symbols were created for the application and they are presented in Figure 4.5

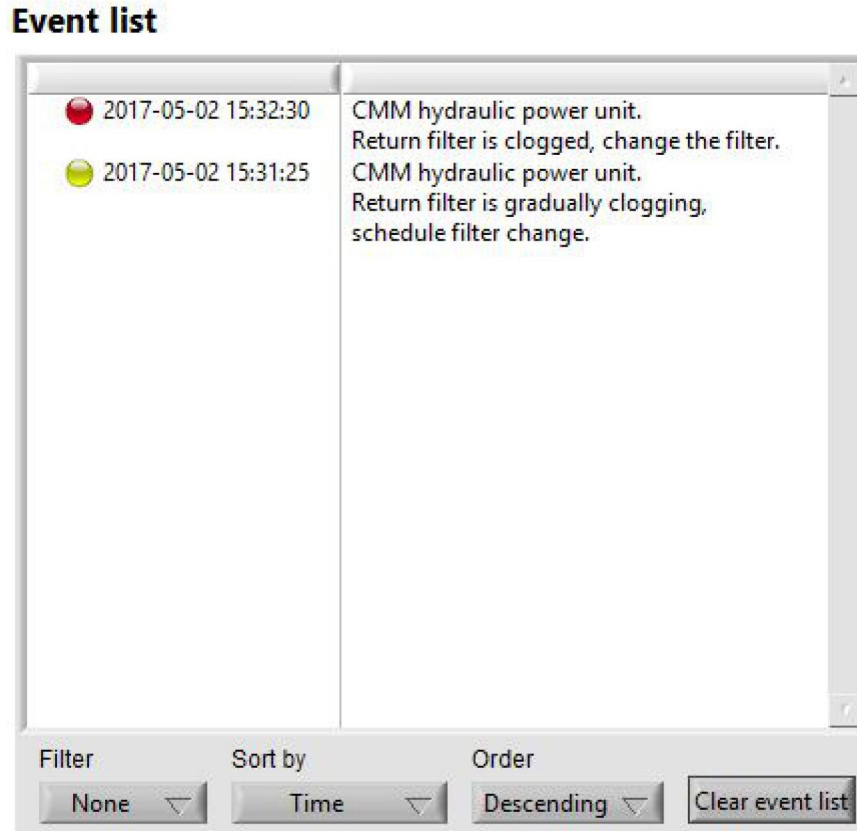


*Figure 4.5* Severity levels for the status monitor and event list. The colors are representing severity levels and symbol types are representing the types of warnings and alarms.

The grey indicator means the equipment is not available or not in use. Green light indicates that everything is running without faults. Yellow indicators are for warnings; arrow down indicates the lower threshold warning and arrow up indicates upper threshold warning. Circular yellow light is general warning symbol for faults, which does not have upper or lower threshold limits. Red indicators are for alarms, and they are used as yellow counterparts. Purple indicator is used if there is something wrong with the signal, for example, if there is a sensor failure or the signal is totally out of range. It is unclear if the purple symbol is needed or not, because the faulty signal can be displayed also as an alarm.

## 4.5 Event Monitor

The event monitor is used like the previously mentioned status monitor, but instead of the statuses, the event monitor collects all the events coming from the system into one monitor. These events includes a severity level, a time stamp and a description of the event. The event monitor of the prototype version is presented in Figure 4.6 and is also visible in the figure in the prototype version in Appendix 1.



**Figure 4.6** The event monitor. The events received from the equipment are displayed here.

The events are created when the status of an equipment, device, component or a primitive changes. The purpose of the event is to notify the operator that something has occurred or there is something the operator should be interested in. The time stamp is created at the same time when the event is created and the message and severity level of the event are created by a primitive. Filtering options for the events is available in order to make reading of the events more simplified.

For more simplified use of the event list, different options for the event list are provided. With the filters it can be selected if all, only warnings or only alarms are displayed. With the sorting option events can be sorted by time, severity level or by event description. With the order option the events can be sorted to descending or ascending order.

## 5. ANALYSIS OF THE REMOTE DIAGNOSTICS APPLICATION PROTOTYPE

The prototype version of the RDA included about half of the features which were in the requirements list. The prototype was meant to be the very first version of the application. In this chapter, the current state of the RDA is reviewed. Also the architecture of the prototype is compared to the ISO 13374 standard, which provided guidelines for the application development process. The future plans for the application are listed in the last section.

### 5.1 Current State (as of May 2017)

In May 17th 2017, the RDA met about 50% of all requirements given for developing the prototype version. Missing features were based mainly on offline mode, which was missing at this point. The offline mode includes history search operations. As soon as the offline mode is fully implemented and integrated to the application, rest of the missing features can be implemented and integrated to the application. The lack of offline features was mostly caused by some problems with implementation process.

Features which were already implemented and integrated to the application, were running without problems. There was some bugs and error handling was incomplete, but for demonstration purposes the application was running fine. There was still some components, which were not very optimised, and those will be updated to better implementations for the new versions of RDA.

A figure of the prototype version is in Appendix 1. In this figure, it is possible to see all the components mentioned in Chapter 4. A signal generator is used to get an example signal to the virtual oscilloscope monitor, which is the largest finished component of the application. Purpose of the prototype was to prove, that the architecture and development decisions used in the application are working as expected and the future development is possible to continue.



### 5.1.1 Demonstration of the Prototype

The prototype version of the RDA was scheduled to be ready in 17th of May. At this day, the customer arrived to VTT to see a live demonstration. It was supposed to give them a live demonstration of the current state of the RDA, and to show the features of the application. The prototype should have included features like data acquisition, use of rules and primitives, and monitoring screens for equipment statuses and event list. In addition, the templates for workbenches and primitives were planned to be a part of the demonstration. With these templates it was demonstrated, how the operators could edit and create new workbenches and primitives by themselves. The previously mentioned offline features were meant to be in this prototype version, but due to limited development resources, this part was left out.

During the demonstration, most of the data was simulated with sine wave created by signal generator. With this simple data, it was possible to present capabilities of the application. Every implemented component of the application were in operation during the demonstration and introduced for the customer. The prototype satisfied the customer expectations at this development phase.

## 5.2 The Prototype Compared to the ISO 13374 Standard

In section 3.6, the architecture guidelines of the standard were shortly reviewed. Figure 3.6 included a reference model of the software architecture and Figure 3.7 included a monitoring reference model based on the provided reference architecture. The prototype version of the RDA can be mapped with the guidelines, and the purpose of this subsection is to compare the front end implementation with the standard.

The front end implementation focuses on state detection and health assessment blocks. The rules and the primitives can be considered as a combination of these two blocks. The primitives are performing state detection operations, which determine if the values are in the warning or in the alarm area. Primitives also are responsible of health assessment, because they include the rules for diagnosing the health of the system. The warnings and alarms are also indicating the diagnostics status.

The RDA has monitoring features for these two blocks, as there is also presented in the standard. Different graphs are provided for the state detection operations. With these, the operator can follow the values visually from the graphs, which can be located in primitive layer. For health assessment, there are indicators for diagnostic

statuses. The statuses of state detection and health assessment are possible to see from the status monitor, which collects all the statuses into one place.

The prototype version of the RDA includes simple prognosis example. It was made for monitoring remaining useful life of a hydraulic filter. This example is presented in the Figure 4.3. In the upcoming versions of the RDA, prognosis will be in more important role. The prototype does not include advisory generation, and therefore there is no recommended actions indicator either.

## 5.3 Future Plans

There is still a lot of things to do with the RDA in order to create high-quality software. Suggestions for improvements and new features were given during the prototype demonstration days. In the following subsections, some of the biggest and most important features and improvements are described in more detail. These subsections are not in the order of importance and are mostly RDA front end implementation related. The future plans of other components of the RDA are not included in this section.

### 5.3.1 Test campaign

The RDA should be tested in-depth in order to test the real performance of the application. For example, there might be bottlenecks or problems with the robustness in the application. The prototype was tested with single signal generator, which itself was running on the same computer as the RDA. This consumes resources of the computer and data transfer over the network is not tested.

There was only this one signal source for the RDA. In reality, the RDA will be receiving the signals from many different sources. For this purpose, it is required to create simulation network, in which several different computers are sending signals to the main computer, in which the RDA is running. The application is made to manage signals and events from 16 different equipment. One idea is to create a test network, which consists of about twenty simulated signal sources. These signal sources could be Raspberry Pi or Beaglebone Black computers. This would be a good test environment and a proper stress test could be done with this kind of equipment. Performance issues and the bottlenecks should be possible to find easily if there are any. [13]

The second test case could include sensors, which are located in different places in



the VTT building. These sensors could measure temperature, moisture and amount of light. These sensors will send data to the RDA, which is tuned for these measures to track these signals. Thresholds, primitives, status monitor and history search could be used and tested by executing this kind of test case. This kind of test case would test the long term stability of the application. [13]

Proper testing of the application will be time consuming. The RDA will be complex and have a lot of separate components. Because of this, finding problems will take a lot of time and it will be difficult to find all the problems before releasing the final version of the RDA.

### 5.3.2 GUI Look-and-Feel Improvements and Web Browser Interface

There was no strict requirements for what the components of the application should look like. The prototype was made with LabVIEW modern graphics palette. Despite the name ‘modern’, the graphics of the prototype are quite simple and they will be switched to silver theme provided by LabVIEW as soon as there is time for the update operation. With the silver theme, the GUI would look better and more like a real final application instead of a prototype version.

There is also a lot of different GUI packages available on the Internet. Modern flat theme is considered to be one option for the next versions, because it would provide more serious and modern look for the application. The default ‘modern’ graphics are practical, but not very fancy or attractive. The silver theme is better, but some graphical components of it are unnecessarily space consuming, and the edges of the indicators are very rounded. It has more modern look compared to the ‘modern’ theme, but the graphics are still a bit too old looking.

The modern flat themes are provided as third-party add-ons, LabVIEW itself does not include them. They are fancy and attractive, but before testing them, it is too early say if they are practical enough for the RDA. The third part add-ons may have some restrictions and all features may not be available. On the other hand, they may provide something what LabVIEW can not provide. Differences between the themes is illustrated in Figure 5.1. The flat theme in the figure is from DMC GUI palette.



**Figure 5.1** Different cancel buttons in LabVIEW. Button on the left is the classic style button, button on center is the silver style button and button on the right is the modern flat style button. The prototype version is made with classic style. The flat style button is third-party add-on provided by DMC.

One of the requirements for developing RDA was a web browser based user interface. Current development tools does not support well creating GUI with web browsers, but the upcoming LabVIEW 2017 version will have capabilities for that. This may require additional work in order to create a new GUI suitable for the web browsers. There is differences between different web browsers, which may cause some difficulties. With a web browser based user interface it is easier to provide the user interface for different computers.

### 5.3.3 More Automatic Application

Currently the RDA configuration file is edited manually, which should be avoided in the future. In the final version of the RDA, almost everything should be done automatically. For example, adding the workbench files should be possible with pressing only one button on the screen. When the operator clicks ‘add new workbench’, the application will add all required files to the configuration file automatically. Currently it is required to add all the files and information manually to the configuration file. When editing this file manually, there might occur user errors. For example, adding one workbench to the configuration file without a memory list of what files should be included to the file is almost impossible, because there are many phases in editing the configuration file. With automation, all required files are added at once without risk of an user error.

This is just one example of many automatic operations of the application. Everything which is possible to do automatically should be changed to work automatically. This saves time and makes using the application more effective. The operators can concentrate on using the application instead of adjusting and tuning it.

The workbench and primitive templates should be very simple. The workbench template is already very simple and easy to use, but the primitive template requires

more attention. One primitive template is not compatible with all kind of diagnostics need in the RDA. In order to provide good templates for different purposes, the example primitives should be created. With this way it is possible to see what components are needed with different types of primitives. When it is learned what the different primitives require, it is possible to create good templates for different purposes. It is not intended that creating new primitives should happen automatically, but it should be as easy as possible for the operators of the application.

## 6. CONCLUSIONS OF THE PROJECT

The aim of the project was to develop a prototype version of the Remote Diagnostics Application for the remote handling systems of ITER. The focus of this thesis was on developing the front end implementation for the application. A new application was developed instead of using ready solutions on the market, because those solutions did not meet the requirements set by ITER at a satisfactory level. The application was developed in co-operation by VTT and TUT.

The application was developed primarily for monitoring the remotely controlled maintenance systems of ITER. The requirements for the customisability of the application caused architectural problems to solve. It was also required, that new rules had to be possible to add to the application without recompiling the whole application. These requirements were covered with the solutions provided by LabVIEW, which was the development platform for the application.

The three-layered architecture was implemented for the application to satisfy these requirements. The architecture was created by using sub panels and dynamic structures, which were available in LabVIEW. The implemented architecture consists of three different layers; the top level layer, the second level ‘workbench’ layer and the third level ‘diagnostics primitive’ layer. This architecture allowed aborting and starting different components of the application without the need of aborting the top level layer of the application. This kind of flexibility covered the requirements for the architecture of the application.

The application also had many other requirements stated by ITER. Some of the requirements were quite loose without strict definitions, how the features should be implemented. This allowed different approaches for implementations of different components of the application.

The prototype version of the Remote Diagnostics Application was demonstrated for the customer in May 2017. The demonstration was successful despite the fact, that some of the features were still under development during the demonstration day. The customer was satisfied with the prototype version and feedback was mostly positive.

The architecture of the application matched well with ISO 13374 standard, and the architecture and the front end implementation were working as expected.


There are still many requirements to cover and new features to implement. Because of the complexity of the application and amount of features, finishing the application will take a lot of time. There is a long list of future plans for the upcoming versions of the application. The development process has been running all the time since the demonstration and new features are already in implementation and integration processes in order to create a new version of the application.

## REFERENCES


- [1] J. Alanen, Senior Research Scientist, VTT.
- [2] J. Alanen, K. Haataja, O. Laurila, J. Peltola, I. Aho, Diagnostics of mobile work machines, VTT Research Notes 2343, VTT Technical Research Centre of Finland, Tampere, 2006, 122 p. Available (accessed 13.9.2017): <http://www.vtt.fi/inf/pdf/tiedotteet/2006/T2343.pdf>.
- [3] A. Avižienis, J.C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing, Vol. 1, Iss. 1, 2004, pp. 11-33.
- [4] Development Life Cycle Models, National Instruments Corporation, web page. Available (accessed 29.08.2017): [http://zone.ni.com/reference/en-XX/help/371361P-01/lvdevconcepts/lifecycle\\_models/](http://zone.ni.com/reference/en-XX/help/371361P-01/lvdevconcepts/lifecycle_models/).
- [5] Electropedia, IEC, web page. Available (accessed 13.7.2017): <http://www.electropedia.org/>.
- [6] Inserting the ship into the bottle, ITER Organization, web page. Available (accessed 20.06.2017): <https://www.iter.org/newsline/singleprint/-/2131>.
- [7] ITER, ITER Organization, web page. Available (accessed 20.06.2017): <https://www.iter.org/>.
- [8] C.K. Mechefske, Machine Condition Monitoring and Fault Diagnostics, in: C.W. de Silva (ed.), Vibration and Shock Handbook, CRC Press, Florida, USA, 2005, pp. 25-1 – 25-35.
- [9] A. Naukkarinen, VTT and TUT major players in the finalization of ITER maintenance robots, web page. Available (accessed 20.06.2017): <http://www.tut.fi/en/about-tut/news-and-events/vtt-and-tut-major-players-in-the-finalization-of-iter-maintenance-robots-p070846c2>.
- [10] M. Niemelä, J. Nurmi, J. Alanen, R. Salokangas, O. Saarela, Selection of Data Analysis Software, VTT, Tampere, unpublished report, 2016, 54 p.
- [11] M. Nyberg, Model Based Fault Diagnosis Methods, Theory, and Automotive Engine Applications, Linköping Studies in Science and Technology. Dissertations No. 591, 1999, 271 p. Available (accessed 13.9.2017): [https://www.fs.isy.liu.se/Publications/PhD/99\\_PhD\\_591\\_MN.pdf](https://www.fs.isy.liu.se/Publications/PhD/99_PhD_591_MN.pdf).

- [12] O. Saarela, Senior Research Scientist, VTT.
- [13] H. Saarinen, Senior Research Scientist, VTT.
- [14] O. Saarinen, J. Alanen, H. Saarinen, J. Nurmi, L. Aha, Requirements and Acceptance Test specification, VTT, Tampere, unpublished report, 2016, 42 p.
- [15] I. Sample, Iter: Flagship fusion reactor could cost twice as much as budgeted, The Guardian, web page. Available (accessed 20.06.2017): <https://www.theguardian.com/science/2009/jan/29/nuclear-fusion-power-iter-funding>.
- [16] SFS-ISO 13374-1, Condition monitoring and diagnostics of machine systems. Data processing, communication and presentation. Part 1: General guidelines, Suomen Standardoimisliitto SFS ry, Helsinki, 2011.
- [17] S. Vassileva, L. Doukovska, V. Sgurev, AI-based Diagnostics for Fault Detection and Isolation in Process Equipment Service, Computing and Informatics, Vol. 33, Iss. 2, pp. 384-409.
- [18] J. Watton, Modelling, Monitoring and Diagnostic Techniques for Fluid Power, Springer, London, 2007, 360 p.
- [19] What is LabVIEW? electronics-notes.com, web page. Available (accessed 20.08.2017): <https://www.electronics-notes.com/articles/test-methods/labview/what-is-labview.php>.
- [20] What Is LabVIEW? National Instruments, web page. Available (accessed 20.08.2017): <http://www.ni.com/en-gb/shop/labview.html>.

# APPENDIX 1. REMOTE DIAGNOSTICS APPLICATION FRONT END IMPLEMENTATION



**Remote Diagnostics Application**



### Equipment statuses

- CMM1
- Hydraulic filter clogging 1
- Radial drive motor currents 1
- CMM 2
- Hydraulic filter clogging 2
- Radial drive motor currents 2
- Equipment 3
- Equipment 4
- Equipment 5
- Equipment 6
- Equipment 7
- Equipment 8
- Equipment 9
- Equipment 10
- Equipment 11
- Equipment 12
- Equipment 13
- Equipment 14
- Equipment 15

Tip-strip: None Update tree

### Event list

- 2017-05-01 15:32:30 CMM hydraulic power unit. CMM filters is clogged, change the filter.
- 2017-05-01 15:31:25 CMM filter is gradually clogging. schedule filter change.

Filter: None Order: Descending Clear event list

Data Acquisition STOP

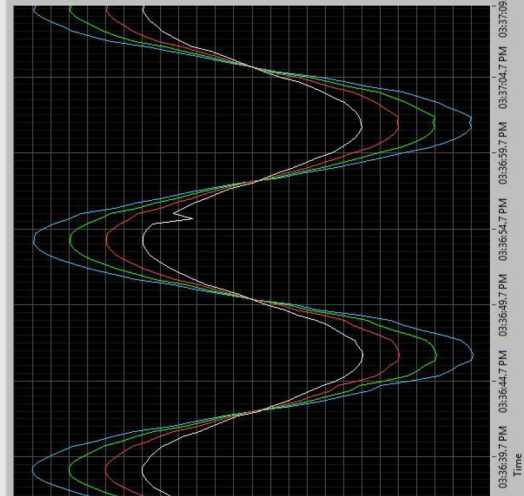
Timespan 1 min

Plot selection Trends

Reset Graphs STOP

Sampling frequency [Hz] 3

Waveform overlay



Tip-strip: None

- EDD Hierarchy
- [-] CMM1 DTP2
- [-] CommNetworkProfile
- [-] NetworksItems
- [-] 31 CMM1 HPU supply and tank filter pressures
- [-] 32 CMM1 HPU supply filter in post pressure
- [-] 33 CMM1 HPU tank filter in post pressure
- [-] 34 CMM1 HPU tank filter out post pressure
- [-] 35 CMM1 test signal1
- [-] 36 CMM1 test signal10
- [-] 37 CMM1 test signal11
- [-] 38 CMM1 test signal12
- [-] 39 CMM1 test signal13
- [-] 40 CMM1 test signal14
- [-] 41 CMM1 test signal15
- [-] 42 CMM1 test signal16
- [-] 43 CMM1 test signal17
- [-] 44 CMM1 test signal18
- [-] 45 CMM1 test signal19
- [-] 46 CMM1 test signal20
- [-] 47 CMM1 test signal21
- [-] 48 CMM1 test signal22
- [-] 49 CMM1 test signal23
- [-] 50 CMM1 test signal24
- [-] 51 CMM1 test signal25
- [-] 52 CMM1 test signal26
- [-] 53 CMM1 test signal27
- [-] 54 CMM1 test signal28
- [-] 55 CMM1 test signal29
- [-] 56 CMM1 test signal30

status code 0 source

Reset Error

STOP APPLICATION

Import new workbenches

Select custom workbenches